



УНИВЕРЗИТЕТ „ГОЦЕ ДЕЛЧЕВ“ – ШТИП
ФАКУЛТЕТ ЗА ИНФОРМАТИКА
ВЕШТАЧКА ИНТЕЛИГЕНЦИЈА И РОБОТИКА
Штип

Владимиров Ангел

**ОДРЕДУВАЊЕ НА ОРИЕНТАЦИЈА НА ПОДВИЖНИ
ОБЈЕКТИ СО ПОМОШ НА ВЕКТОРСКИ МЕРЕЊА ОД
ВИДЕО КАМЕРА**

– МАГИСТЕРСКИ ТРУД –

Штип, Септември, 2019 година

Комисија за оценка и одбрана

Ментор: Сашо Коцески
Проф. д-р, Факултет за информатика
Универзитет „Гоце Делчев“ Штип

Член: Владо Гичев
Проф. д-р, Факултет за информатика
Универзитет „Гоце Делчев“ Штип

Член: Васко Кокаланов
Проф. д-р, Факултет за информатика
Универзитет „Гоце Делчев“ Штип

Датум на одбрана: _____

Датум на промоција: _____

Рецензирани и објавени трудови

1. Attitude Determination of Unmanned Aerial Vehicle Using Single Camera Vector Observations - International Journal of Computer Applications (0975 – 8887) Volume 178 – No. 41, August 2019

ОДРЕДУВАЊЕ НА ОРИЕНТАЦИЈА НА ПОДВИЖНИ ОБЈЕКТИ СО ПОМОШ НА ВЕКТОРСКИ МЕРЕЊА ОД ВИДЕО КАМЕРА

КРАТОК ИЗВАДОК:

Овој магистерски труд презентира една рамка за одредување на ориентацијата на подвижни објекти со помош на векторски мерења од единечна камера. Претпоставуваме дека имаме позната околина во форма на мапа и точни позиции на подвижен објект од каде мерењата се земени. Итеративни нумерички решенија базирани на Гаус-Њутновиот метод и Левенберг-Маркарт методот се презентирани.

КЛУЧНИ ЗБОРОВИ:

Одредување на ориентација/ Гаус-Њутнов метод/ Левенберг-Маркартов метод/ Матлаб-(lsqnonlin)/ калибрација на камера/

ATTITUDE DETERMINATION OF MOVING OBJECTS FROM CAMERA VECTOR OBSERVATIONS

ABSTRACT:

In this master thesis is presented proper framework for attitude determination of a moving rigid body from single camera vector observations in a known environment. We assume that we have known environment in a form of a map with exact positions of the moving object from where the measurement are taken. Iterative numerical results based on Gauss-Newton method and Levenberg-Marquardt method are presented.

KEY WORDS:

*Attitude determination/ Gauss Newton method/
Levenberg-Marquardt method/ Matlab-(lsqnonlin)/
Camera calibration/*

Содржина:

1. Вовед	8
2. Репрезентација на движење во три-димензионален простор	9
2.1. Три димензионален Еуклидски простор	9
2.2. Движење на круто тело	13
2.3. Ротациско движење и негово претставување	18
2.3.1. Ортогонални матрици за репрезентација на ротации	18
2.3.2. Канонични експоненцијални координати за ротација	21
2.4. Приказ на движења на круто тело	23
2.4.1. Хомогена репрезентација	25
2.4.2. Канонични експоненцијални координати за движење на круто тело	27
2.5. Трансформации на координати и брзини	28
3. Компјутерска визија	32
3.1. Формирање на слика	32
3.1.1. Репрезентација на слики	32
3.1.2. Леќи, светлина и основна фотометрија	35
3.1.2.1. Репрезентација на слика преку леќи	35
3.1.2.2. Репрезентација на слика без леќа	37
3.1.3. Геометриски модел за формирање на слика	39
3.1.3.1. Идеална перспективна камера	40
3.1.3.2. Камера со внатрешни параметри	41
3.1.3.3. Радијална дисторзија	45
3.2. Процесирање на слика	47
3.2.1. Точкасти оператори	48
3.2.1.1. Трансформација на пиксел	50
3.2.1.2. Трансформации на боја	51
3.2.1.3. Составување и поставување подлога	52
3.2.1.4. Изедначување на хистограм	53
3.2.2. Линеарно филтрирање	55
3.2.3. Нелинеарно филтрирање	60
3.2.4. Пирамиди на слика	63
3.2.5. Геометриски трансформации	69
3.3. Одредување на значајки	74
3.3.1. Точки	74
3.3.1.1. Детектори на значајки	75
3.3.1.2. Опис на значајки	79

3.3.2	Краеви и кошиња.....	81
3.3.2.1	Детекција на краеви и кошиња.....	81
3.3.3	Линии.....	85
3.3.3.1	Последователни апроксимации.....	86
3.3.3.2	Хугх трансформација.....	86
4.	Computer Vision Toolbox во Матлаб.....	90
4.1	Главни карактеристики.....	90
4.2	Одредување на карактеристични точки (значајки).....	91
4.2.1	Локални значајки.....	92
4.2.2	Употреба на значајки.....	93
4.3	Калибрирање на камера.....	94
5.	Методи на оптимизација базирани на градиент.....	99
5.1	Услови за оптималност.....	100
5.2	Алгоритми.....	100
6.	Одредување на ориентација со помош на векторски мерења од единечна видео камера.....	104
6.1	Геометриски модел на единечна камера.....	104
6.2	Равенки за одредување на ориентација.....	105
7.	Практични експерименти и резултати.....	108
7.1	Примена на Гаус-Њутновиот метод и Левенберг-Маркартовиот метод за одредување на ориентација.....	109
7.2	Резултати од практичните експерименти.....	114
8.	Заклучок и работа во иднина.....	118
9.	Користена литература.....	120
	Додаток 1. Однос помеѓу Ојлерови агли и кватерниони.....	125

1. Вовед

Одредување (пресметување) на став или ориентација на подвижни објекти е од голема важност во денешно време како во секојдневното живеење така и во многу сектори во индустријата. Во филмската индустрија се користи за компјутерско генерирање на слики и ликови. Во автомобилската индустрија се користи како алатка за инспекција и мониторирање на делови. Во одбраната се користи за следење и управување на подвижни објекти, а воедно и како заштита на безбедноста на сопствените сили од непријателски подвижни објекти. Во секторот на медицината се користи за рехабилитирање на повреди на телото кај човекот. Додека кај спортот може да придонесе до пронаоѓање начин како да се постигнат подобри резултати. За одредување на ориентацијата на подвижните објекти се користат различни видови на сензори. Денес со напредувањето на технологијата често се користат MEMS инерцијалните сензори (анг. MEMS), кои што се мали, ефтини и прецизни. Сепак видео сензорите се еден друг вид на сензори, кои што денес се задолжителни за употреба. Тие обезбедуваат богат извор на информации дадени како релативни мерења помеѓу параметрите на подвижниот објект (позиција, брзина и ориентација) и околината. Овој магистерски труд презентира една рамка за одредување на ориентацијата на подвижни објекти со помош на векторски мерења од единечна камера. Претпоставуваме дека имаме позната околина во форма на мапа и точни позиции на подвижниот објект од каде мерењата се земени. Итеративни нумерички решенија базирани на Гаус-Њутновиот метод и Левенберг-Маркарт методот се презентирани. Предностите и недостатоците на двете решенија се презентирани.

2. Репрезентација на движење во три-димензионален простор

Проучувањето на геометриската поврзаност помеѓу три-димензионалната сцена и дво-димензионалните слики, кои што се направени од камера во движење, претставува меѓусебна интеракција на две фундаментални множества на трансформации: **Еуклидско движење**, или така наречено движење на круто тело, кое што има за цел да проектира (моделира) како се движи камерата и **проекција на слика**, што го објаснува процесот на формирање на слика. Уште пред овие две трансформации да се соединат во компјутерската визија, тие се поединечно развиени. Механиката има голем придонес во основата за проучувањето на принципите за движење на тела. Во ова поглавје, ќе се запознаеме со три-димензионалниот Еуклидски простор како и со движењата на круто тело. Во наредното поглавје ќе се фокусираме на проекцијата на слика, односно моделот на камерата [1].

2.1. Три димензионален Еуклидски простор

За да го означиме Еуклидскиот простор ќе ја употребуваме ознаката \mathbb{E}^3 . Во главно, Еуклидскиот простор е множество, чии што елементи ги исполнуваат петте аксиоми на Еуклид [2]:

- 1) “Да може да се нацрта линија која минува од една точка во друга точка.”
- 2) “Правата линија да нема почеток и крај.”
- 3) “Сите прави агли да се исти еден со друг.”
- 4) “Да се опише круг околу било која точка со одреден радиус.”
- 5) “Претпоставка за паралелни линии: ако две прави пресекуваат трета права и при тоа збирот на внатрешните агли од едната страна на првата, е помал од 180, тогаш двете прави мора да се пресечат една со друга доколку се продолжат доволни долго.”

Три димензионалниот Еуклидски простор може да биде претставен глобално преку координатниот систем, така да секоја точка $p \in \mathbb{E}^3$ може да се идентификува со точка во \mathbb{R}^3 со три координати.

$$X = [X_1, X_2, X_3]^T = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} \in \mathbb{R}^3 \quad (1)$$

Понекогаш, може да се употребува и $[X, Y, Z]^T$ за да се означат одредени координати наместо $[X_1, X_2, X_3]^T$. Преку таква претпоставка за Еуклидскиот простор и координатниот систем, вршме кореспонденција помеѓу \mathbb{E}^3 и \mathbb{R}^3 , што овозможува слободно да зборуваме за точки и нивните координати како да се едно исто.

Со ова, координатите од координатниот систем ни овозможуваат мерења на должини и агли [1]. За да се овозможи тоа, \mathbb{E}^3 треба да биде претставен во метри.

Дефиниција (вектор): Во Еуклидскиот простор, вектор v е одреден со две точки $p, q \in \mathbb{E}^3$ и е означен со стрелка која ги поврзува точките p и q , означен како $v = \overrightarrow{pq}$.

Точката p се нарекува почетна точка на векторот v . Во координати векторот v е претставен со $[v_1, v_2, v_3]^T \in \mathbb{R}^3$, каде што секоја координата претставува разликата помеѓу кореспондентните координати од двете точки: ако p има координати X и q има координати Y , тогаш v има координати

$$v = Y - X \in \mathbb{R}^3. \quad (2)$$

Претходната дефиниција за вектор е наменета за така наречен *врзан вектор*. Исто така постои и *слободен (неврзан) вектор*, вектор за чија што дефиниција не е потребна почетната точка. Ако имаме два пара точки (p, q) и (p', q') кои што го исполнуваат условот $Y - X = Y' - X'$, тогаш велиме дека тие го дефинираат истиот слободен вектор. Преку ова својство се овозможува векторот v да биде претставен било каде во \mathbb{E}^3 . Посебно за овој пример со горе наведеното, може да се заклучи дека почетната точка на векторот е всушност координатниот почеток на координатниот систем, така што $X = 0$ и $Y = v$. Само да се појасни оваа нотација: Y овде ги претставува координатите на вектор, чии што координати се исти како на точката q , само затоа што ја имаме избрано точката p да биде во координатниот почеток. Треба да се напомене дека точките и векторите се различни геометриски објекти. Ова е важно, бидејќи движењето на круто тело различно влијае врз точки, а различно врз вектори.

Множеството од сите слободни вектори формира така наречен *линеарен векторски простор*, со линеарна комбинација од два вектора $v, u \in \mathbb{R}^3$ дефинирана како:

$$\alpha v + \beta u = [\alpha v_1 + \beta u_1, \alpha v_2 + \beta u_2, \alpha v_3 + \beta u_3]^T, \quad \forall \alpha, \beta \in \mathbb{R}. \quad (3)$$

Еуклидските метрички единици за \mathbb{E}^3 се дефинирани од скаларниот производ од векторскиот простор \mathbb{R}^3 . Со правилен избор во координатниот систем, било кој скаларен производ во \mathbb{E}^3 може да биде изменет во следнава стандардна (канонична) форма

$$\langle u, v \rangle \doteq u^T v = u_1 v_1 + u_2 v_2 + u_3 v_3, \quad \forall u, v \in \mathbb{R}^3. \quad (4)$$

Овој скаларен производ е исто така познат како стандардна Еуклидска мерна единица. Во поголем дел од овој магистерски труд ќе ја употребуваме каноничната (стандардната) форма на скаларниот производ $\langle u, v \rangle = u^T v$. Следствено нормализација на вектор v е $\|v\| \doteq \sqrt{\langle u, v \rangle} = \sqrt{v_1^2 + v_2^2 + v_3^2}$. Кога скаларниот производ помеѓу два вектора е нула, т.е. $\langle u, v \rangle = 0$, тогаш за тие два вектора се вели дека се *ортогонални*.

Значи, Еуклидскиот простор \mathbb{E}^3 , во поглед на координатниот систем, формално може да се опише дека претставува простор \mathbb{R}^3 со мерни единици добиени од скаларскиот производ. Со такви мерни единици, може да се мери не само должина помеѓу две точки или агли помеѓу вектори, исто така може да се пресмета и должината на крива или волумен на регион.

Додека при скаларен производ на два вектора се добива резултат скалар, при *векторски производ* на два вектора се добива резултат вектор, дефиниран подолу.

Дефиниција (векторски производ). На два вектора $u, v \in \mathbb{R}^3$, нивниот векторски производ е трет вектор со координати добиени од:

$$u \times v \doteq \begin{bmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{bmatrix} \in \mathbb{R}^3. \quad (5)$$

Директно од оваа дефиниција е изведено дека векторскиот производ на два вектора е линеарен во секој свој аргумент:

$$u \times (\alpha v + \beta w) = \alpha u \times v + \beta u \times w, \forall \alpha, \beta \in \mathbb{R}. \quad (6)$$

Следува дека

$$\langle u \times v, u \rangle = \langle u \times v, v \rangle = 0, \quad u \times v = -v \times u. \quad (7)$$

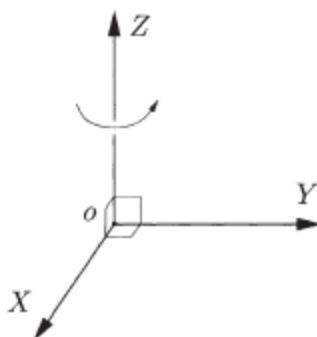
Затоа следи дека векторскиот производ на два вектора е ортогонален на секој од векторите поединечно, а редот на векторите ја одредува ориентацијата (односно ако се измени редот на векторите, векторскиот производ го менува знакот).

Ако го наместиме u , векторскиот производ може да биде претставен како мапа од \mathbb{R}^3 во $\mathbb{R}^3: v \mapsto u \times v$. Мапата е линеарна во v и затоа може да биде претставена како матрица. Оваа матрица ја означуваме со $\hat{u} \in \mathbb{R}^{3 \times 3}$, и се означува како “ u капа”. Согласно ова следува верификација на матрицата:

$$\hat{u} \doteq \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix} \in \mathbb{R}^{3 \times 3}. \quad (8)$$

Оттука можеме да напишеме $u \times v = \hat{u}v$. Напомена дека \hat{u} е 3×3 косо симетрична матрица, т.е. $\hat{u}^T = -\hat{u}$.

Пример 1 (Правило на десна рака). Потврдуваме дека за $e_1 = [1, 0, 0]^T$, $e_2 = [0, 1, 0]^T \in \mathbb{R}^3$ добиваме $e_1 \times e_2 = [0, 0, 1]^T = e_3$. Во стандардниот координатен систем, ова претставува векторски производ на основните оски X и Y при што се добива основната оска Z . Поради ова векторскиот производ го потврдува правилото на десна рака, види Слика 1



Слика 1. Десно-ориентиран (X, Y, Z) координатен систем.

Векторскиот производ е природно дефинира мапа помеѓу вектор u и 3×3 косо симетрична матрица \hat{u} . За проверка на оваа, лесно можеме да идентификуваме

три-димензионален вектор поврзан со секоја 3×3 косо симетрична матрица (само треба u_1, u_2, u_3 да се изведат според равенката (2.2)).

Лема 1 (Косо симетрична матрица). Матрица $M \in \mathbb{R}^{3 \times 3}$ е косо симетрична ако и само ако $M = \hat{u}$ за некое $u \in \mathbb{R}^3$.

Поради тоа, векторскиот простор \mathbb{R}^3 и просторот на сите косо симетрични 3×3 матрици, наречен $so(3)$, се изоморфни (т.е. постои еден-на-еден мапа со што се зачувува структурата на векторскиот простор). Изоморфизмот е така наречен “капа оператор”

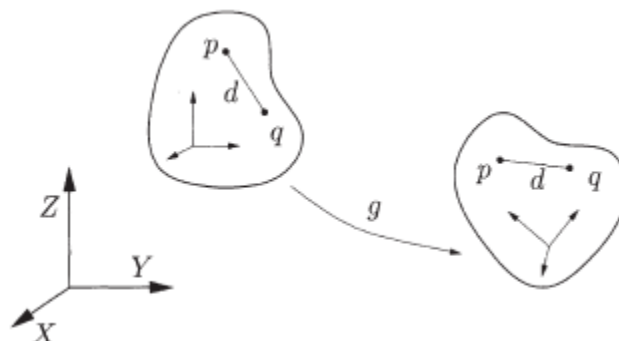
$$\Lambda: \mathbb{R}^3 \rightarrow so(3); u \mapsto \hat{u} \quad (9)$$

а неговата спротивна (инверзна) мапа е наречена “вее оператор”, кој што ги определува компонентите на вектор u од косо симетрична матрица \hat{u} , е даден како

$$V: so(3) \rightarrow \mathbb{R}^3; \hat{u} \mapsto \hat{u}^V = u. \quad (10)$$

2.2. Движење на круто тело

Да замислиме дека даден објект се движи пред камера. Со цел да го објасниме движењето на објектот, треба да се определи траекторијата на движење на секоја точка од тој објект. Сепак кај круто тело нема потреба да се определи движењето на секоја негова точка. Доволно е да се определи движењето на една точка и движењето на три координатни оски со почеток во таа точка. Причината за тоа е следнава. Кај секое круто тело растојанието меѓу две точки кои се наоѓаат на него, во текот на времето нема да се промени како што ќе има промена во движењето на самото тело, види Слика 2 [1]:



Слика 2. Движење на круто тело и зачувувањето на растојанието d помеѓу било кој пар на точки (p, q) на него.

Значи ако $X(t)$ и $Y(t)$ се координатите на било кои две точки p и q на телото, следи дека растојанието помеѓу нив е константно:

$$\|X(t) - Y(t)\| \equiv \text{constant}, \forall t \in \mathbb{R}. \quad (11)$$

Движење на круто тело (или трансформација на круто тело) претставува група на мапи што опишуваат како координатите на секоја точка од крутото тело, се менува со текот на време при тоа задоволувајќи го условот (11). Таквата мапа ја означуваме како:

$$g(t): \mathbb{R}^3 \rightarrow \mathbb{R}^3; X \mapsto g(t)(X). \quad (12)$$

Доколку се концентрираме на мапата помеѓу почетната и конечната конфигурација, а не целосната патека на телото што се движи, ќего добиеме поместувањето на крутото тело, означено како :

$$g: \mathbb{R}^3 \rightarrow \mathbb{R}^3; X \mapsto g(X). \quad (13)$$

Освен трансформација на координатите на точките, g вклучува и трансформација на векторите. Да претпоставиме дека v е вектор дефиниран со две точки p и q со координати $v = Y - X$; тогаш по трансформацијата g добиваме нов вектор

$$v = g_*(v) = g(Y) - g(X). \quad (14)$$

Бидејќи g го зачувува растојанието помеѓу точките, имаме $\|g_*(v)\| = \|v\|$ за сите слободни вектори $v \in \mathbb{R}^3$.

Мапа која што го зачувува растојанието се нарекува Еуклидска трансформација. Во 3-D просторот, множеството од сите Еуклидски трансформации се означува со $E(3)$. Треба да напоменеме дека задржувањето на растојанието меѓу точки не е доволно за да се окарактеризира движењето на крутото тело во просторот. Всушност постојат трансформации што го задржуваат растојанието, а сепак не се физички изводливи. На пример мапата

$$f: [X_1, X_2, X_3]^T \mapsto [X_1, X_2, -X_3]^T \quad (15)$$

го зачувува растојанието, но не и ориентацијата. Врши одсликување на точки во XY-рамнината како на двојно огледало. За да се користат правилно мапите, потребно е за секое движење на круто тело да се зачувуваат: растојанието меѓу точки и ориентацијата. Тоа значи дека дополнително на зачувувањето на нормализацијата на векторите, мора исто така да се зачувува и нивниот векторски производ. Мапата или трансформацијата вклучена во движењето на круто тело се нарекува *специјална Еуклидска трансформација*. Зборот специјална го означува фактот дека трансформацијата ја задржува и ориентацијата (насоката) [1].

Дефиниција (Движење на круто тело или специјална Еуклидска трансформација). Мапа $g: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ претставува движење на круто тело или специјална еуклидска трансформација ако ги задржува (зачувува) нормализацијата и векторскиот производ на било кои два вектори,

1. Нормализација: $\|g * (v)\| = \|v\|, \quad \forall v \in \mathbb{R}^3,$
2. Векторски производ: $g * (u) \times g * (v) = g * (u \times v), \quad \forall u, v \in \mathbb{R}^3.$

Збирот на сите такви движења или трансформации се означува со $SE(3)$.

Во погоре наведената дефиниција за движења на круто тело, не е веднаш воочливо дека аглиите помеѓу векторите се зачувуваат. Како и да е, скаларниот производ $\langle \cdot, \cdot \rangle$ може да се претстави во условите на нормализација $\| \cdot \|$ со т.н. поларизационен идентитет

$$\langle u, v \rangle = \frac{1}{4}(\|u + v\|^2 - \|u - v\|^2) \quad (16)$$

бидејќи $\|u + v\| = \|g_*(u), g_*(v)\|$ можеме заклучиме дека за секое движење на круто тело g ,

$$\langle u, v \rangle = \langle g_*(u), g_*(v) \rangle, \quad \forall u, v \in \mathbb{R}^3. \quad (17)$$

Со други зборови движењето на круто тело може да се објасни како движење со кое се зачувуваат скаларниот и векторскиот производ [1].

Пример (производ од три вектори и волумен). Од дефиницијата за движење на круто тело, може да се забележи дека се задржува и така наречен производ од три вектори:

$$\langle g_*(u), g_*(v) \times g_*(w) \rangle = \langle u, v \times w \rangle \quad (18)$$

Оттука, бидејќи производот на трите вектори е еднаков на волуменот што го зафаќа паралелопипедот формиран од трите вектори, следи дека движењето на круто тело исто така го зачувува и волуменот.

Како овие својства на движењето на круто тело ни овозможуваат попрецизно да го објасниме движењето? Својството да се зачувува растојанието и ориентацијата на точките од телото, значи дека точките не можат да се движат во зависност една од друга. Како последица на ова, движење круто тело може да се опише само преку движење на една точка која се наоѓа на телото и ротацијата на координатниот систем кој што е врзан за таа точка.

Со цел да се види подобро ова, се прикажува *конфигурацијата* на круто тело врзана за координатен систем со почеток во една точка од самото тело, и понатаму ќе го следиме движењето на целиот овој врзан координатен систем на телото во споредба со координатен систем врзан за Земјата (Земјен координатен систем).

Да замислиме даден координатен систем кој што како 3-оски има 3 нормални вектори $e_1, e_2, e_3 \in \mathbb{R}^3$; и ги исполнуваат условите:

$$e_i^T e_j = \delta_{ij} \doteq \begin{cases} 1 & \text{for } i = j, \\ 0 & \text{for } i \neq j. \end{cases} \quad (19)$$

Векторите се подредени така да формираат десен координатен систем (со правилото на десна рака): $e_1 \times e_2 = e_3$. Потоа по движење на круто тело g , имаме

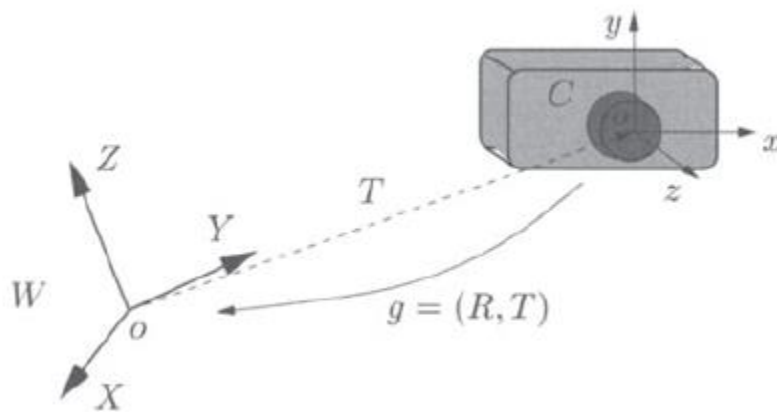
$$g_*(e_i)^T g_*(e_j) = \delta_{ij}, \quad g_*(e_1) \times g_*(e_2) = g_*(e_3). \quad (20)$$

Тоа се трите нормални еден на друг вектори $g_*(e_1), g_*(e_2), g_*(e_3)$ кои сеуште го формираат десниот координатен систем. Така круто тело секогаш ќе биде претставено со десен координатен систем, кој што систем се нарекува координатен систем врзан за телото или само врзан координатен систем додека движење на телото ќе биде претставено со движењето на тој координатен систем.

На Слика 3 е претставен објект, односно камера, која што се движи релативно на Земјен координатен систем $W: (X, Y, Z)$. Со цел да се покаже

конфигурацијата на камерата во споредба со Земјениот координатен систем, ќе избереме фиксна точка o на камерата за која што ќе вземе координатниот систем т.н координатен систем на камерата, $C: (x, y, z)$. Кога камерата се движи, заедно со неа се движи и координатниот систем на камерата. Конфигурацијата на камерата е определена со две компоненти:

1. Векторот помеѓу центарот o на Земјениот координатен систем и центарот на координатниот систем на камерата $g(o)$, наречен уште *транслационен дел* кој се означува со T ;
2. Ориентацијата на координатниот систем на камерата C , со координатни оски (x, y, z) , релативен на фиксниот Земјен координатен систем W , со координатни оски (X, Y, Z) , наречен уште *ротирачки дел* означен со R .



Слика 3. Движење на круто тело (камера) претставено со координатен систем на камерата $C: (x, y, z)$ и координатен систем на Земјата $W: (X, Y, Z)$.

Во проблемите што ќе ги опфатиме овде, можеме да бираме каде ќе биде почетокот на Земјениот координатниот систем. Дали ќе биде врзан за камерата па ќе ги одредуваме транслациите и ротациите од сцената релативни на камерата, или пак почеток ќе биде врзан за некоја точка на сцената, па да ќе го одредуваме движењето на камерата релативно на тој координатниот систем на сцената. Едноставно е важно да го имаме соодносот на движење измеѓу камерата и сцената.

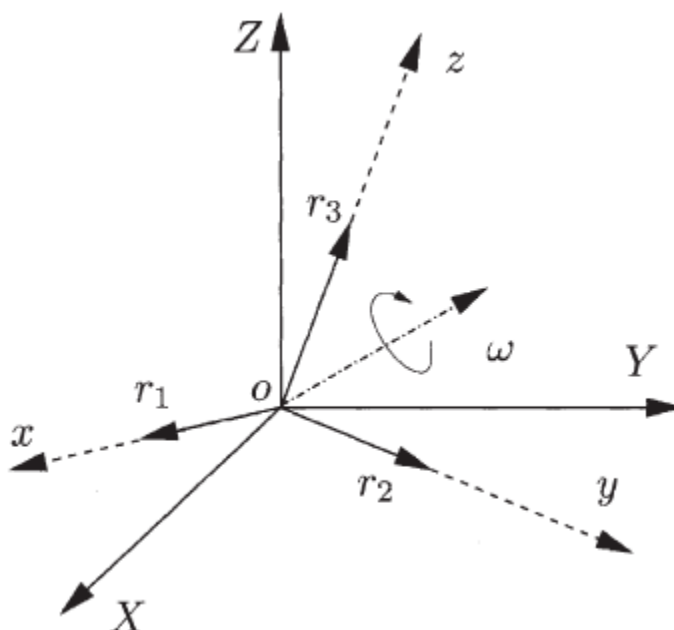
Ако можеме да движеме круто тело како камерата, од едно место до друго, секако дека ќе можеме и да го вратиме процесот во почетната состојба. Слично

преку комбинација на неколку движења може да генерираме и ново движење. Ова својство на враќање и соединување, математички може да се нарече како *група*. Како што ќе видиме и понатаму множеството на движења на круто тело е всушност група, така наречена Еуклидска група [1].

2.3. Ротациско движење и негово претставување

2.3.1. Ортогонални матрици за репрезентација на ротации

Да претпоставиме дека имаме круто тело кое што ротира околу фиксна точка $o \in \mathbb{E}^3$. Како ја опишуваме неговата ориентација релативна на координатниот систем W што сме го избрале? Секогаш можеме да претпоставиме дека центарот на координатниот систем на Земјата е центарот на ротацијата o . Доколку центарот на ротација не е во центарот на координатниот систем, едноставно преку translација може да се доведе тие две точки да се преклопат. Сега да додадеме уште еден координатен систем C , исто така со почеток во центарот o , што би претставувало ротирачки објект, да речеме камера. Релацијата помеѓу овие два координатни системи е прикажана во Слика 4 [1].



Слика 4. Ротација на круто тело околу фиксна точка o и околу оската ω . Координатниот систем W (означен со полна линија) е фиксен, а

координатниот систем C (означен со испрекината линија) е врзан на круто тело што ротира.

Конфигурацијата или ориентацијата на координатниот систем C во однос на координатниот систем W е одредена од страна на координатите на трите нормални вектори $r_1 = g_*(e_1), r_2 = g_*(e_2), r_3 = g_*(e_3) \in \mathbb{R}^3$ релативни на координатниот систем на Земјата W , како што е прикажано на Слика 4. Трите вектори r_1, r_2, r_3 се едноставно нормализирани вектори кои соодветствуваат од трите оски x, y, z на координатниот систем C . Конфигурацијата на ротирачкиот објект е целосно претставена со 3×3 матрица

$$R_{wc} \doteq [r_1, r_2, r_3] \in \mathbb{R}^{3 \times 3} \quad (21)$$

Бидејќи r_1, r_2, r_3 формираат нормален координатен систем, следи дека

$$r_i^T r_j = \delta_{ij} \doteq \begin{cases} 1 & \text{for } i = j, \\ 0 & \text{for } i \neq j, \end{cases} \quad \forall i, j \in \{1, 2, 3\}. \quad (22)$$

Ова можеме да го запишеме во форма на матрици како

$$R_{wc}^T R_{wc} = R_{wc} R_{wc}^T = I. \quad (23)$$

Матрицата која што го задоволува овој услов е *ортогонална матрица*. Од дефиницијата погоре следи дека инверзна матрица на ортогоналната матрица е всушност транспонирана матрица: $R_{wc}^{-1} = R_{wc}^T$. Бидејќи r_1, r_2, r_3 формираат десен координатен систем, понатаму имаме услов да детерминантата на R_{wc} биде $+1$. Поради тоа R_{wc} е *специјална ортогонална матрица*, а како што спомнавме и претходно зборот специјална укажува на задржување на ориентацијата. Просторот од сите специјални ортогонални матрици во $\mathbb{R}^{3 \times 3}$ се означува како:

$$SO(3) \doteq \{R \in \mathbb{R}^{3 \times 3} | R^T R = I, \det(R) = +1\}. \quad (24)$$

Традиционално, 3×3 специјално ортогоналните матрици уште се нарекуваат и *матрици на ротирање*. Може да се потврди дека $SO(3)$ ги задоволува сите четири аксиоми за група во множење на матрици. Просторот $SO(3)$ исто се нарекува и *специјална ортогонална група* од \mathbb{R}^3 , или кратко група на ротирање. Директно од дефиницијата може да се докаже дека ротациите навистина ги зачувуваат скаларниот и векторскиот производ на вектори [1].

Пример (матрица на ротација).

Матрица која што прикажува ротација околу Z -оската за агол θ е :

$$R_Z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (25)$$

Исто така може да се изведат и матриците за ротација околу X -оската и Y -оската [1].

Се враќаме на Слика 4, секоја матрица на ротирање $R_{wc} \in SO(3)$ претставува можна конфигурација на објект ротиран околу точка o . Освен тоа, R_{wc} има уште една улога како матрица која што ги прикажува трансформациите на координатите од координатниот систем C во координатниот систем W . Да претпоставиме дека за дадена точка $p \in \mathbb{E}^3$, нејзините координати прикажани во координатниот систем W се $X_w = [X_{1w}, X_{2w}, X_{3w}]^T \in \mathbb{R}^3$. Бидејќи r_1, r_2, r_3 исто така формираат основа за \mathbb{R}^3 , X_w може да се претстави како линеарна комбинација од овие три вектора, како $X_w = X_{1c}r_1 + X_{2c}r_2 + X_{3c}r_3$ со $[X_{1c}, X_{2c}, X_{3c}]^T \in \mathbb{R}^3$. Очигледно дека $X_c = [X_{1c}, X_{2c}, X_{3c}]^T$ се координатите на истата точка p само релативни на координатниот систем C . Затоа имаме

$$X_w = X_{1c}r_1 + X_{2c}r_2 + X_{3c}r_3 = R_{wc}X_c. \quad (26)$$

Во оваа равенка, матрицата R_{wc} ги трансформира координатите X_c од точката p релативни на координатниот систем C во координати X_w релативни на координатниот систем W . Бидејќи R_{wc} е матрица на ротација, нејзината инверзна матрица е всушност транспонираната матрица на таа матрица,

$$X_c = R_{wc}^{-1}X_w = R_{wc}^T X_w. \quad (27)$$

Инверзираната трансформација од ротацијата е исто така ротација; ја нарекуваме R_{cw} , по што следи

$$R_{cw} = R_{wc}^{-1} = R_{wc}^T. \quad (28)$$

Конфигурацијата на постојано ротиран објект може да се објасни како траекторија $R(t): t \mapsto SO(3)$ во просторот $SO(3)$. Кога почетното време не е $t = 0$, движењето помеѓу време t_2 и време t_1 ќе биде означено како $R(t_2, t_1)$. Композициониот закон на групата за ротација подразбира [1]:

$$R(t_2, t_0) = R(t_2, t_1)R(t_1, t_0), \forall t_0 < t_1 < t_2 \in \mathbb{R}. \quad (29)$$

За камера која ротира, земјените координати X_w за некоја фиксна 3-Д точка p се трансформирани во координати релативни на координатниот систем на камерата C преку

$$X_c(t) = R_{cw}(t)X_w. \quad (30)$$

Алтернативно доколку точката p е фиксна релативно на координатниот систем на камерата со координати X_c , тогаш нејзините координати од земјениот координатен систем $X_w(t)$ во функција од времето t се дадени како [1]:

$$X_w(t) = R_{wc}(t)X_c. \quad (31)$$

2.3.2. Канонични експоненцијални координати за ротација

До сега прикажавме дека ротацијата на круто тело во \mathbb{E}^3 може да биде претставено преку 3×3 матрица на ротација $R \in SO(3)$. Во матричниот приказ што го имавме до сега, секоја матрица на ротација R е опишана со помош на своите $3 \times 3 = 9$ члена. Овие 9 члена не се слободни параметри, бидејќи тие мораат да го исполнат ограничувањето $R^T R = I$. Ова всушност задава шест независни ограничувања на 9-те члена. Па отука следи дека големината на матриците за ротација $SO(3)$, треба да биде само со три члена, а шест члена од 9-те се всушност непотребни. Во ова поглавје ќе дефинираме неколку основни параметри за големината на матриците на ротација [1].

Дадена патека $R(t): \mathbb{R} \rightarrow SO(3)$ која што опишува постојаното ротациско движење, мора да ги исполни следниве услови:

$$R(t)R^T(t) = I. \quad (32)$$

Одредувањето на изводот на претходната равенка, во зависност од времето t и со забелешката дека десната страна од равенката е константа, добиваме

$$\dot{R}(t)R^T(t) + R(t)\dot{R}^T(t) = 0 \Rightarrow \dot{R}(t)R^T(t) = -(\dot{R}(t)R^T(t))^T. \quad (33)$$

Добиената матрица покажува на фактот дека матрицата $\dot{R}(t)R^T(t) \in \mathbb{R}^{3 \times 3}$ е косо симетрична матрица така што мора да постои вектор $\omega(t) \in \mathbb{R}^3$ таков што

$$\dot{R}(t)R^T(t) = \hat{\omega}(t). \quad (34)$$

Ако ги помножиме двете страни со $R(t)$

$$\dot{R}(t) = \hat{\omega}(t)R(t). \quad (35)$$

Од горе наведенава равенка, доколку $R(t_0) = I$ за $t = t_0$, имаме $R(t_0) = \hat{\omega}(t_0)$. Оттука, со единечната матрица I , косо симетрична матрица дава апроксимација од прв ред на матрицата на ротација:

$$R(t_0 + dt) \approx I + \hat{\omega}(t_0)dt. \quad (36)$$

И како што и очекувавме, големината на косо симетричните матрици е одредена како

$$so(3) \doteq \{\hat{\omega} \in \mathbb{R}^{3 \times 3} | \omega \in \mathbb{R}^3\}, \quad (37)$$

И во продолжение на опсервацијата погоре, истата се нарекува и *тангентен простор* на идентитетот на групата на ротации $SO(3)$. Ако $R(t)$ не е во идентитетот, тангентниот простор на $R(t)$ е едноставно $so(3)$ пренесен во $R(t)$ со множење со $R(t)$ од десно: $\dot{R}(t) = \hat{\omega}(t)R(t)$. Ова исто така покажува дека, елементите од $SO(3)$ зависат само од три параметри $(\omega_1, \omega_2, \omega_3)[1]$.

Со претходо изнесеното за локална апроксимација, да се обидеме да најдеме корисна репрезентација на матрицата на ротација. Да претпоставиме дека матрицата $\hat{\omega}$ во (35) е константа,

$$\dot{R}(t) = \hat{\omega}R(t). \quad (38)$$

Во равенката горе, $R(t)$ може да се интерпретира како *главна преносна матрица* за следнава линеарна диференцијална равенка (ODE):

$$\dot{x}(t) = \hat{\omega}x(t), \quad x(t) \in \mathbb{R}^3 \quad (39)$$

каде важно е веднаш да се потврди решението на оваа равенка дека е

$$x(t) = e^{\hat{\omega}t}x(0), \quad (40)$$

каде $e^{\hat{\omega}t}$ е експонентот на матрицата

$$e^{\hat{\omega}t} = I + \hat{\omega}t + \frac{(\hat{\omega}t)^2}{2!} + \dots + \frac{(\hat{\omega}t)^n}{n!} + \dots \quad (41)$$

Експонентот $e^{\hat{\omega}t}$ е често означен со $\exp(\hat{\omega}t)$. Поради уникатноста на решението на диференцијалната равенка (39), и претпоставувајќи дека $R(0) = I$ е почетниот услов за (38), мора да имаме

$$R(t) = e^{\hat{\omega}t}. \quad (42)$$

За да потврдиме дека матрицата $e^{\hat{\omega}t}$ е навистина матрица на ротација, може директно да покажеме од дефиницијата за експонентот на матрицата дека

$$e^{\hat{\omega}t^{-1}} = e^{-\hat{\omega}t} = e^{\hat{\omega}^T t} = (e^{\hat{\omega}t})^T. \quad (43)$$

Од тука $(e^{\hat{\omega}t})^T e^{\hat{\omega}t} = I$. Останува да се види дека $\det(e^{\hat{\omega}t}) = +1$ [1].

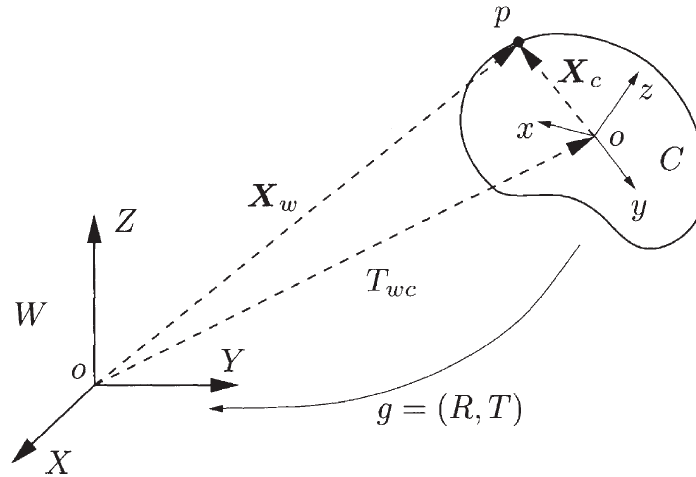
Физичка интерпретација на равенката (42) е ако $\|\omega\| = 1$, тогаш $R(t) = e^{\hat{\omega}t}$ е едноставно ротација околу оската $\omega \in \mathbb{R}^3$ за агол од t радијани. Генерално t може да биде апсорбирано внатре во ω , така да ќе имаме $R = e^{\hat{\omega}}$ за ω со произволна нормализација. Следи дека експонентот на матрицата (41) точно дефинира мапа од просторот $so(3)$ до $SO(3)$ така наречена *експоненцијална мапа*

$$\exp: so(3) \rightarrow SO(3); \quad \hat{\omega} \mapsto e^{\hat{\omega}}. \quad (44)$$

Дојдовме до равенката (42) со претпоставка дека $\omega(t)$ во (35) е константа. Сепак ова не сегокаш случај. Прашање кое природно се поставува е дали може секоја матрица на ротација $R \in SO(3)$ да биде претставена во експоненцијална форма како во равенка (42). Одговорот е да и фактот е потврден во [1].

2.4. Приказ на движења на круто тело

Во претходниот дел од ова поглавје, дискутиравме само за ротација на круто тело и како да ја претставиме и пресметаме матрица на ротација. Во ова поглавје ќе дискутираме како да претставиме движење на круто тело во општо, движење со ротација и транслација.



Слика 5 Движење на круто тело помеѓу подвижен координатен систем C и глобалниот координатен систем W .

Сликата 5 илустрира движење на круто тело прикажан со координатен систем C . За да ги најдеме координатите на точка p од објектот во однос на глобалниот координатниот систем W , јасно се гледа на сликата дека векторот X_w претставува збир транслации $T_{wc} \in \mathbb{R}^3$ од координатниот почеток на координатниот систем C релативен на центарот од глобалниот координатен систем W и векторот X_c прикажан во однос на глобалниот координатен систем W . Бидејќи X_c се координатите на точката p релативни во однос на координатниот систем C . Преминот во координатниот систем W е со $R_{wc}X_c$, каде што $R_{wc} \in SO(3)$ е матрицата на ротацијата помеѓу двата координатни системи. Па отука координатите X_w се дадени како

$$X_w = R_{wc}X_c + T_{wc}. \quad (45)$$

Целото движење на крутото тело се претставува со $g_{wc} = (R_{wc}, T_{wc})$, или поедноставно со $g = (R, T)$, ако не се земат во обзир координатните системи. Тогаш g претставува не само опис на конфигурацијата на крутото тело, туку и трансформација на координатите помеѓу двата координатни система. Во компактна форма тоа се запишува како

$$X_w = g_{wc}(X_c). \quad (46)$$

Множеството од сите можни конфигурации на круто тело може да биде опишано од просторот за движење на крути тела или од специјално Еуклидски трансформации

$$SE(3) = \{g = (R, T) | R \in SO(3), T \in \mathbb{R}^3\}. \quad (47)$$

Забелешка дека $g = (R, T)$ не е матрица за $SE(3)$. За да се здобие со приказ на матрица потребно е да се запознаеме со така наречени *хомогени координати*.

2.4.1. Хомогена репрезентација

Можеме да се забележиме од равенката (45), во споредба со чистата равенка за ротација, трансформацијата на координатите за целосно движење на круто тело, не претставува линеарна, туку поместена трансформација (анг. Affine transformation). За да ја претвориме поместената трансформација во линеарна трансформација, применуваме хомогени координати. Додаваме 1 на координатите $X = [X_1, X_2, X_3]^T \in \mathbb{R}^3$ на точка $p \in \mathbb{E}^3$ да додаде вектор во \mathbb{R}^4 , означен како

$$\bar{X} \doteq \begin{bmatrix} X \\ 1 \end{bmatrix} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{bmatrix} \in \mathbb{R}^4. \quad (48)$$

Такво додавање на координати, го вградува Еуклидскиот простор \mathbb{E}^3 во хиперамнина \mathbb{R}^4 , наместо \mathbb{R}^3 . Хомогените координати на вектор $v = X(q) - X(p)$ се дефинираат како разликата меѓу хомогени координати на две точки од следнава форма

$$\bar{v} \doteq \begin{bmatrix} v \\ 0 \end{bmatrix} = \begin{bmatrix} X(q) \\ 1 \end{bmatrix} - \begin{bmatrix} X(p) \\ 1 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 0 \end{bmatrix} \in \mathbb{R}^4. \quad (49)$$

Забелешка дека во \mathbb{R}^4 , вектори од горе наведената форма, даваат зголемување на простор, а сите линеарни структури на векторите $v \in \mathbb{R}^3$ се одлично зачувани со новиот приказ. Со употребата на новиот приказ, поместената трансформација (45) може да се запише во линеарна форма

$$\bar{X}_w \doteq \begin{bmatrix} X_w \\ 1 \end{bmatrix} = \begin{bmatrix} R_{wc} & T_{wc} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ 1 \end{bmatrix} \doteq \bar{g}_{wc} \bar{X}_c, \quad (50)$$

Каде што 4x4 матрицата $\bar{g}_{wc} \in \mathbb{R}^{4 \times 4}$ се нарекува *хомогена репрезентација* на движење на круто тело $\bar{g}_{wc} = (R_{wc}, T_{wc}) \in SE(3)$. Во главно, ако $g = (R, T)$, тогаш хомогената репрезентација би била

$$\bar{g} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{4 \times 4}. \quad (51)$$

За избегнување на редундантност во ознаките и со исфрлање на вишокот во симболи, може да се прикаже трансформација на координати на круто тело со линеарно множење на матрици. Хомогената репрезентација на g во (51) овозможува зголемување на природниот изглед на матрицата од специјалните Еуклидски трансформации

$$SE(3) \doteq \left\{ \bar{g} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \mid R \in SO(3), T \in \mathbb{R}^3 \right\} \subset \mathbb{R}^{4 \times 4}. \quad (52)$$

Употребувајќи ја оваа репрезентација, лесно може да се покаже дека множеството $SE(3)$ навистина ги задоволува сите потреби на група. Особено за $\forall g_1, g_2$ и $g \in SE(3)$, каде имаме

$$\bar{g}_1 \bar{g}_2 = \begin{bmatrix} R_1 & T_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_2 & T_2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_1 R_2 & R_1 T_2 + T_1 \\ 0 & 1 \end{bmatrix} \in SE(3) \quad (53)$$

и

$$\bar{g}_1^{-1} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} R^T & -R^T T \\ 0 & 1 \end{bmatrix} \in SE(3). \quad (54)$$

Затоа, \bar{g} е навистина приказ на матрица за групата на движења на крути тела според дефиницијата која што ја спомнавме во поглавје 2.2. Во хомогениот приказ, движењето на крутото тело $g \in SE(3)$ околу вектор $v = X(q) - X(p) \in \mathbb{R}^3$ постанува

$$\bar{g}_*(\bar{v}) = \bar{g}\bar{X}(q) - \bar{g}\bar{X}(p) = \bar{g}\bar{v}. \quad (55)$$

Тоа е движењето е едноставно прикажано со множење на матрици. Во 3-D координатите, имаме $g_*(v) = Rv$, бидејќи само деловите од ротација влијаат врз векторите. Може да се потврди дека таква акција само ги зачувува скаларниот и векторскиот производ. Како што може да се забележи, дека движењата влијаат различно врз точки (транслација и ротација), а различно врз вектори (само ротација) [1].

2.4.2. Канонични експоненцијални координати за движење на круто тело

Во претходното поглавје дискутиравме за експоненцијални координати за матрици на ротација $R \in SO(3)$. Слично координати постојат и за хомогенатата репрезентација за целосно движење на круто тело $g \in SE(3)$ [1].

Да претпоставиме дека постојаното движење на круто тело е опишано како траекторија на $SE(3)$: $g(t) = (R(t), T(t))$, или претставено со хомогена репрезентација

$$g(t) = \begin{bmatrix} R(t) & T(t) \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{4 \times 4}. \quad (56)$$

Од сега па натаму, за поедноставување кога и да има двозначност, ќе го избегнуваме знакот " – " со цел да иницираме хомогена репрезентација и едноставно да користиме g . Ќе ги употребуваме истите ознаки за точки, X за \bar{X} , и за вектори, v за \bar{v} .

Со случајот на чиста ротација, ајде прво да ја погледнеме структурата на матрицата

$$\dot{g}(t)g^{-1}(t) = \begin{bmatrix} \dot{R}(t)R^T(t) & \dot{T}(t) - \hat{\omega}(t)T(t) \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{4 \times 4}. \quad (57)$$

Од проучувањето на матрицата на ротација забележуваме дека $\dot{R}(t)R^T(t)$ е косо симетрична матрица; т.е. постои $\hat{\omega}(t) \in so(3)$ таков да $\hat{\omega}(t) = \dot{R}(t)R^T(t)$.

Дефинираме вектор $v(t) \in \mathbb{R}^3$ таков да $v(t) = \dot{T}(t) - R^T(t)T(t)$, тогаш горната равенка постанува

$$\dot{g}(t)g^{-1}(t) = \begin{bmatrix} \hat{\omega}(t) & v(t) \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{4 \times 4}. \quad (58)$$

Ако понатаму дефинираме матрица $\hat{\xi} \in \mathbb{R}^{4 \times 4}$ да биде

$$\hat{\xi}(t) = \begin{bmatrix} \hat{\omega}(t) & v(t) \\ 0 & 0 \end{bmatrix}, \quad (59)$$

од каде имаме

$$\dot{g}(t) = (\dot{g}(t)g^{-1}(t))g(t) = \hat{\xi}(t)g(t), \quad (60)$$

Каде што на $\hat{\xi}$ може да се гледа како тангентен вектор долж кривата на $g(t)$ и може да се искористи за пресметување на $g(t)$ локално:

$$g(t + dt) \approx g(t) + \hat{\xi}(t)g(t)dt = (I + \hat{\xi}(t)dt)g(t). \quad (61)$$

4x4 матрица формирана од $\hat{\xi}$ се нарекува твист (анг. twist). Множеството на сите криви се означува како

$$se(3) \doteq \left\{ \hat{\xi} = \begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix} \mid \hat{\omega} \in so(3), v \in \mathbb{R}^3 \right\} \subset \mathbb{R}^{4 \times 4}. \quad (62)$$

Множеството $se(3)$ се нарекува тангентен простор (или Ли алгебра) од групата матрици $SE(3)$. Исто така дефинираме и два оператора " \vee " и " \wedge " за конвертирање помеѓу твистот $\hat{\xi} \in se(3)$ и неговите координати на твист $\xi \in \mathbb{R}^6$:

$$\begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix}^{\vee} \doteq \begin{bmatrix} v \\ \omega \end{bmatrix} \in \mathbb{R}^6, \quad \begin{bmatrix} v \\ \omega \end{bmatrix}^{\wedge} \doteq \begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{4 \times 4}. \quad (63)$$

Во координатите на твистот ξ , v ќе претставува линеарна брзина, а ω ќе претставува аголна брзина, што кажува дека тие се поврзани или со транслација или со ротација од целосно движење [1].

2.5. Трансформации на координати и брзини

Често доаѓаме во ситуација кога треба да знаеме како и колку координатите и брзината на една точка се менуваат кога камерата се движи. Ова е така бидејќи е поочигледно да го избереме координатниот систем на камерата како референтен систем, па да ги објасниме движењата на камерата и 3-D точките релативни на камерата. Бидејќи камерата можеби е во движење, мора да знаеме како да трансформираме величини како координати и брзини од еден координатен систем на камерата во друг. Или со други зборови треба да знаеме како точно да ја опишеме позицијата и брзината на точка релативна на камерата што се движи.

Правила на трансформација на координати

Времето $t \in \mathbb{R}$ обично ќе биде користено како индекс за движењето на камерата. Дури и во случај во кои само слики се дадени, ќе го земеме времето t да биде

индекс на позицијата на камерата и сликата во тој момент. Затоа ќе го користиме $g(t) = (R(t), T(t)) \in SE(3)$ или пак

$$g(t) = \begin{bmatrix} R(t) & T(t) \\ 0 & 1 \end{bmatrix} \in SE(3) \quad (64)$$

За да го означиме релативното поместување меѓу глобалниот координатен систем W и координатниот систем на камерата C во време $t \in \mathbb{R}$. тука ќе го игнорираме долниот индекс cw од записот $g_{cw}(t)$ се додека е разбирлива поентата [1]. Претпоставуваме дека $g(0) = I$ односно во време $t = 0$ координатниот систем на камерата се совпаѓа со глобалниот координатниот систем. Па ако координатите на точка $p \in \mathbb{E}^3$, релативно на глобалниот координатен систем се $X_0 = X(0)$, а релативно на камерата во дадено време t координатите се

$$X(t) = R(t)X_0 + T(t), \quad (65)$$

или пак претставени во хомогена репрезентација

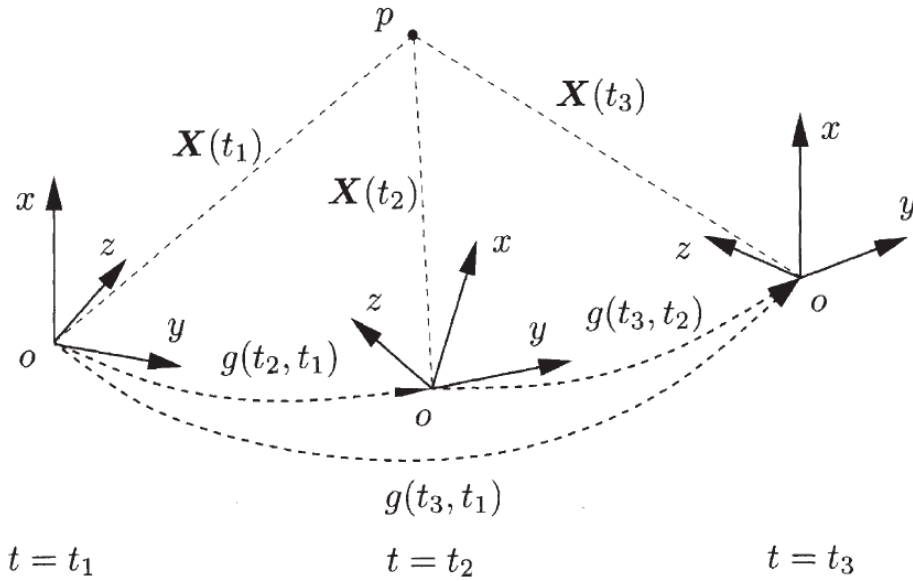
$$X(t) = g(t)X_0. \quad (66)$$

Ако камерата се наоѓа на позиција $g(t_1), g(t_2), \dots, g(t_m)$ во време t_1, t_2, \dots, t_m соодветно, тогаш координатите на точката p се дадени како $X(t_i) = g(t_i)X_0, i = 1, 2, \dots, m$. Ако е важна само позицијата, а не и времето, употребуваме g_i како кратенка за $g(t_i)$ и слично R_i за $R(t_i)$, T_i за $T(t_i)$, и X_i за $X(t_i)$. Следи дека

$$X_i = R_iX_0 + T_i. \quad (67)$$

Кога почетното време не е $t = 0$, движењето на камерата меѓу време t_2 и време t_1 ќе се означи како $g(t_2, t_1) \in SE(3)$. Тогаш го имаме следниов однос помеѓу координати на иста точка, но во различни времиња:

$$X(t_2) = g(t_2, t_1)X(t_1), \quad \forall t_1, t_2 \in \mathbb{R}. \quad (68)$$



Слика 6. Композиција на движење на круто тело. $X(t_1), X(t_2), X(t_3)$ се координатите на точката p во однос на трите координатини системи на камерата во време $t = t_1, t_2, t_3$ соодветно.

Сега да претпоставиме трета позиција на камерата во $t = t_3 \in \mathbb{R}$, како што е прикажано на Слика 6. Релативното движење помеѓу камерата во t_3 и t_2 е $g(t_3, t_2)$, а помеѓу t_3 и t_1 е $g(t_3, t_1)$. Па отука ги имаме следниве релации меѓу координатите:

$$X(t_3) = g(t_3, t_2)X(t_2) = g(t_3, t_2)g(t_2, t_1)X(t_1). \quad (69)$$

Споредено со директната релација помеѓу координатите во t_3 и t_1 ,

$$X(t_3) = g(t_3, t_1)X(t_1), \quad (70)$$

следствено мора да важи следново правило

$$g(t_3, t_1) = g(t_3, t_2)g(t_2, t_1). \quad (71)$$

т.н правилото на составување (анг. composition rule) кое ги означува координатите X на точката p релативно на било која позиција каде што се наоѓа камерата, ако координатите се познати во однос на одредена позиција. Ова правило го опфаќа и правилото на инверзија

$$g^{-1}(t_2, t_1) = g(t_1, t_2) \quad (72)$$

Од каде $g(t_2, t_1)g(t_1, t_2) = g(t_2, t_2) = I$. Во случаи каде времето не е од значење, ги користиме g_{ij} како кратенка за $g(t_i, t_j)$. Тогаш правилото на составување во хомоген приказ е

$$X_i = g_{ij}X_j, g_{ik} = g_{ij}g_{jk}, g_{ij}^{-1} = g_{ji}. \quad (73)$$

Правила на трансформација на брзина

Откако ја проучивме трансформацијата на координати, сега ќе видиме како таа влијае врз брзината. Знаеме дека координатите $X(t)$ на точка $p \in \mathbb{E}^3$ релативни на подвижна камера се функции од времето t :

$$X(t) = g_{cw}(t)X_0 \quad (74)$$

Тогаш брзината на точката p релативна на координатниот систем на камерата е

$$\dot{X}(t) = \dot{g}_{cw}(t)X_0 \quad (75)$$

Со цел да го изразиме $\dot{X}(t)$ како величина во подвижниот координатен систем, го заменуваме X_0 со $g_{cw}^{-1}(t)X(t)$ и користејќи ја забелешката од кружното движење [1], дефинираме

$$\hat{V}_{cw}^c(t) = \dot{g}_{cw}(t)g_{cw}^{-1}(t) \in se(3) \quad (76)$$

Каде што изразот за $\dot{g}_{cw}(t)g_{cw}^{-1}(t)$ може да се најде во (57). Формулата (75) може да се презапише како

$$\dot{X}(t) = \hat{V}_{cw}^c(t)X(t) \quad (77)$$

Бидејќи \hat{V}_{cw}^c има форма

$$\hat{V}_{cw}^c(t) = \begin{bmatrix} \hat{w}(t) & v(t) \\ 0 & 0 \end{bmatrix}, \quad (78)$$

Можеме да ја запишеме равенката за брзина на точка во 3-D координати, наместо во хомогени координати, како

$$\dot{X}(t) = \hat{w}(t)X(t) + v(t). \quad (79)$$

Физичкиот запис на симболот \hat{V}_{cw}^c е всушност брзината на движење на глобалниот координатен систем релативен на координатниот систем на камерата. За да се јасно објасни физичкиот запис за брзина, мораме да

напоменеме кој координатниот систем се движи во споредба на кој координатен систем, и од кој координатен систем се гледа. Ако ја смениме позицијата од која што ја надбљудуваме брзината, равенката ќе се смени исто така. На пример, да претпоставиме дека гледачот е во друг координатен систем кој е изместен со помош на трансформација релативно на координатниот систем на камерата $g \in SE(3)$. Тогаш координатите на истата точка p релативно на координатниот систем на гледачот се $Y(t) = gX(t)$. Ја пресметуваме брзината во новиот координатен систем и добиваме

$$\dot{Y}(t) = g\dot{g}_{cw}(t)g_{cw}^{-1}(t)g^{-1}Y(t) = g\hat{V}_{cw}^c g^{-1}Y(t). \quad (80)$$

Така да новата равенка на брзина е

$$\hat{V} = g\hat{V}_{cw}^c g^{-1}. \quad (81)$$

Ова е истата физичка величина, но прикажана од друга гледана точка. Гледаме дека двете брзини се поврзани преку мапирање од релативното движење g .

$$ad_g: se(3) \rightarrow se(3); \xi \mapsto g\xi g^{-1}. \quad (82)$$

Ова е така наречено *мапа на поврзување* на просторот $se(3)$. Употребувајќи ги овие забелешки во претходниот пример ќе добиеме $\hat{V} = ad_g(\hat{V}_{cw}^c)$. Види како мапата на поврзување ја трансформира брзината од еден координатен систем во друг. Употребувајќи го фактот дека $g_{cw}(t)g_{cw}^{-1}(t) = I$, јасно може да се потврди дека

$$\hat{V}_{cw}^c = \dot{g}_{cw}g_{cw}^{-1} = -g_{cw}^{-1}\dot{g}_{cw} = -g_{cw}(\dot{g}_{cw}g_{cw}^{-1})g_{cw}^{-1} = ad_{g_{cw}}(-\hat{V}_{wc}^w). \quad (83)$$

Од тука \hat{V}_{cw}^c исто така може да се објасни како *негативна брзина* на камерата која што се движи релативно на глобалниот координатен систем, гледано во координатниот систем на камерата [1].

3. Компјутерска визија

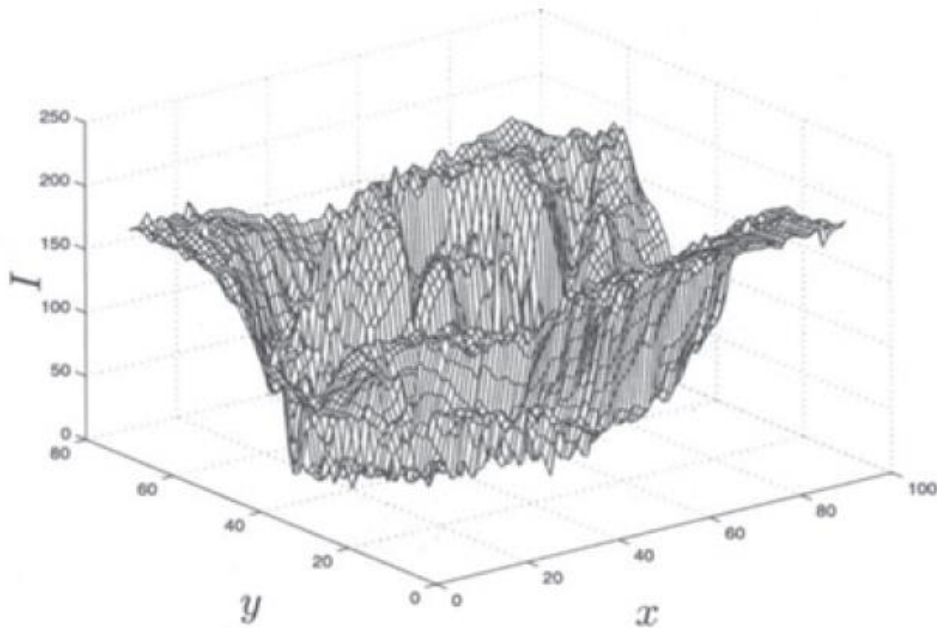
3.1 Формирање на слика

3.1.1 Репрезентација на слики

Зборот слика, во компјутерска смисла, претставува дводимензионална низа, чиешто елементи се вредностите за интензитетот на бојата (анг.RGB) нијанси од црвена, сина и зелена, доколку сликата е во боја или доколку е црно бела тогаш тоа се нијанси од сива боја) [1]. Со други зборови, претставува мапа I , дефинирана на компактен регион Ω , на дводимензионална површина која може да прима само реални позитивни броеви. На пример ако се работи за камера, Ω е правоаголна рамнина добиена од фотографски медиум како CCD сензор. I претставува функција

$$I: \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}_+; (x, y) \mapsto I(x, y). \quad (84)$$

Таква слика (функција) може да биде претставена со помош на график од I како што е претставено во слика 7.



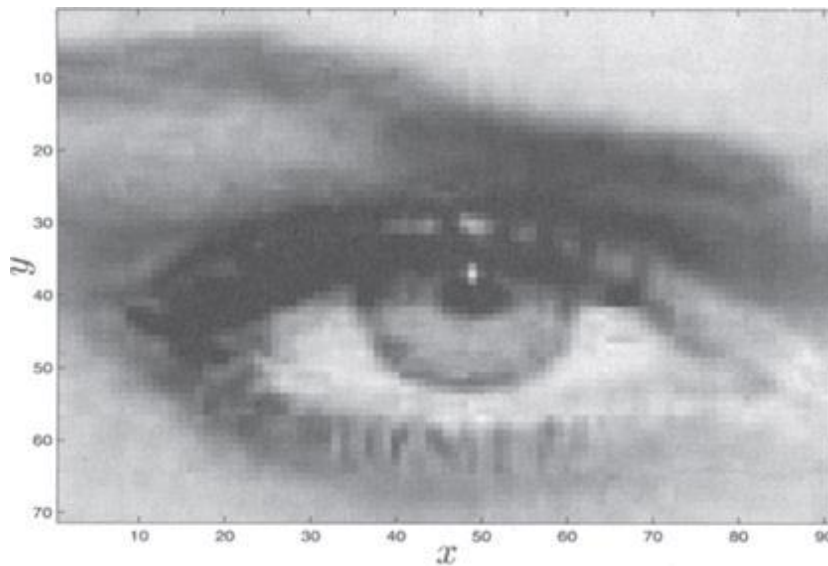
Слика 7. Слика I претставена како дводимензионална рамнина, графикон на I .

Во случај на дигитална слика рамнината Ω и опсегот \mathbb{R}_+ се дискретизирани. На пример $\Omega = [1, 640] \times [1, 480] \subset \mathbb{Z}^2$, а \mathbb{R}_+ е одреден со интервал од вредностите $[0, 255] \subset \mathbb{Z}_+$. таква слика може да се прикаже со низа од броеви како во Слика 7[1].

188	186	188	187	168	130	101	99	110	113	112	107	117	140	153	153	156	158	156	153
189	189	188	181	163	135	109	104	113	113	110	109	117	134	147	152	156	163	160	156
190	190	188	176	159	139	115	106	114	123	114	111	119	130	141	154	165	160	156	151
190	188	188	175	158	139	114	103	113	126	112	113	127	133	137	151	165	156	152	145
191	185	189	177	158	138	110	99	112	119	107	115	137	140	135	144	157	163	158	150
193	183	178	164	148	134	118	112	119	117	118	106	122	139	140	152	154	160	155	147
185	181	178	165	149	135	121	116	124	120	122	109	123	139	141	154	156	159	154	147
175	176	176	163	145	131	120	118	125	123	125	112	124	139	142	155	158	158	155	148
170	170	172	159	137	123	116	114	119	122	126	113	123	137	141	156	158	159	157	150
171	171	173	157	131	119	116	113	114	118	125	113	122	135	140	155	156	160	160	152
174	175	176	156	128	120	121	118	113	112	123	114	122	135	141	155	155	158	159	152
176	174	174	151	123	119	126	121	112	108	122	115	123	137	143	156	155	152	155	150
175	169	168	144	117	117	127	122	109	106	122	116	125	139	145	158	156	147	152	148
179	179	180	155	127	121	118	109	107	113	125	133	130	129	139	153	161	148	155	157
176	183	181	153	122	115	113	106	105	109	123	132	131	131	140	151	157	149	156	159
180	181	177	147	115	110	111	107	107	105	120	132	133	133	141	150	154	148	155	157
181	174	170	141	113	111	115	112	113	105	119	130	132	134	144	153	156	148	152	151
180	172	168	140	114	114	118	113	112	107	119	128	130	134	146	157	162	153	153	148
186	176	171	142	114	114	116	110	108	104	116	125	128	134	148	161	165	159	157	149
185	178	171	138	109	110	114	110	109	97	110	121	127	136	150	160	163	158	156	150

Слика 8 Слика *I* претставена како дводимензионална матрица

Вредностите на сликата *I* зависат од физичките карактеристики од сцената која што се гледа, од формата, дистрибуцијата на светлината, рефлектирачките карактеристики. И покрај тоа што на слика 7 и слика 8 не се забележуваат карактеристиките, односно не се симболично претставени карактеристиките кои ги има сцената што се проектира, сепак тоа е начинот на кој што тие се претставуваат во компјутерот. Различно претставување на истата слика со подобра можност за истакнување на карактеристиките кај човечкото око, е добиено преку генерирање на слика. Сликата може да биде перципирана, од страна на човечкото око, малку различна отколку каква што е реалноста. Всушност сликата претставува *контролирана илузија* на реалноста[1]: постојат сцени различни од реалноста (бидејќи сликата е рамна). Слика 9 е истата слика *I* која е прикажана на слика 7 и слика 8, Иако на втората слика 9 изгледа дека има повеќе информации за карактеристиките на сцената, но и покрај тоа што е друга репрезентација на формата на сликата, сепак ги содржи точно истите карактеристики за сликата.



Слика 9. Фотографија од слика I .

3.1.2 Леќи, светлина и основна фотометрија

Со цел да го објасниме процесот на формирање на слика, мора да ја дефинираме вредноста на $I(x, y)$ што се доделува во секоја точка (x, y) во Ω . Таква вредност $I(x, y)$ се нарекува *интензитет на слика* или формално *ирадијанса* (анг. irradiance). Единица мерка која се користи е енергија на метар квадратен (W/m^2) и претставува енергија која паѓа на еден мал дел од фотографскиот сензор [1].

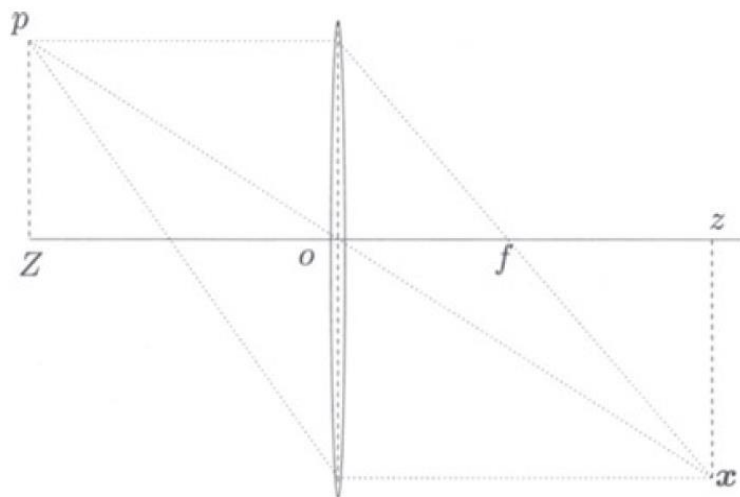
3.1.2.1 Репрезентација на слика преку леќи

Камерата е составена од систем од леќи кои служат за да ја насочуваат светлината. Под насочување на светлина се мисли контролирана промена на во насоката на пропагирање (растење), која што може да се добие преку дефракција, рефракција или рефлексija. За поедноставување ги занемаруваме ефектите на дефракција и рефлексija во леќите, а ќе ја земеме во предвид само рефракцијата (прекршување на зраци). Ќе обратиме внимание само на најпростиот пример за рефракција, односно на моделот на *тенка леќа*.

Тенка леќа, слика 10, е математички модел дефиниран со оска, наречена *оптичка оска*, и рамнина нормална на оската, наречена *фокална рамнина* со

кружен отвор во оптичкиот центар, т.е. пресекот на фокалната рамнина со оптичката оска. Тенката леќа има две вредности: првата е тоа што сите зраци кои што влегуват во отворот паралелно на оската, самата оска ја пресекуваат на растојание f од оптичкиот центар, наречено фокално растојание на леќата. Втората вредност на леќата е тоа што сите зраци кои минуваат низ центарот не се превртуваат. Да поставиме точка $p \in \mathbb{E}^3$, не многу далеку од оптичката оска, на далечина Z долж оптичката оска до оптичкиот центар. Да нацртаме два зрака од точката p : еден паралелен на оската, и еден што минува во центарот (слика 10). Првиот зрак ја пресекува оптичката оска кај фокусот, а вториот зрак останува непроменет. Да ја наречеме точката x местото каде двата зрака ќе се пресечат, а z да е растојанието од оптичкиот центар. Со разложување на секој друг зрак од точката p , во два зрака, еден зрак паралелен на оптичката оска и еден директно во оптичкиот центар, можеме да заклучиме дека сите зраци од p се пресекуваат во x на спротивната страна од леќата. Така да зрак од x паралелен на оската мора да помине во точката p . Така од слика 10 може да ја добиеме следнава *фундаментална равенка на тенка леќа* [1].

$$\frac{1}{Z} + \frac{1}{z} = \frac{1}{f} \quad (85)$$



Слика 10. Пресликување на точката p е точката x , односно пресекот на зраците кои одат паралелно на оптичката оска со зраците кои минуваат во оптичкиот центар.

Точката x ќе се нарекува пресликување на точката p . Поради тоа ирадијансата $I(x)$ во точката x со координати (x, y) , е добиена со емитирање на целосната енергија од просторот во рамнината на сликата, преку геометријата на леќата.

3.1.2.2 Репрезентација на слика без леќа

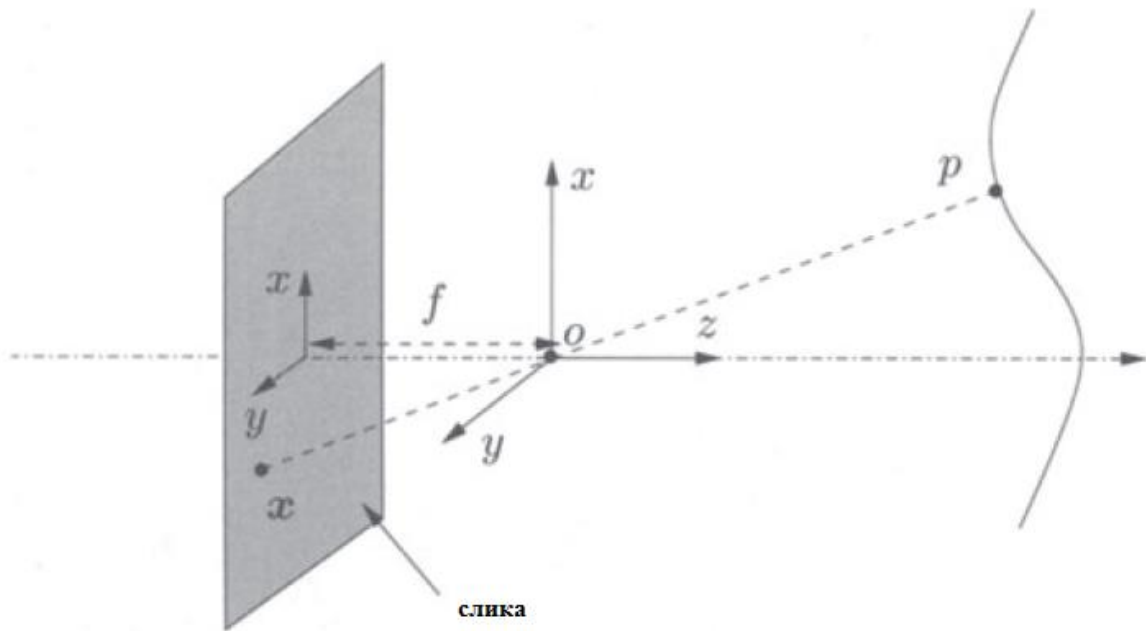
Ако прекршувањето на зраците го сведиме на минимум (нула), сите зраци ќе бидат фокусирани да минуваат низ оптичкиот центар и ќе останат не прекршени. Како последица на тоа единствените точки кои ќе влијаат на ирадијансата во точката на пресликување $x = [x, y]^T$, се наоѓаат на линијата што минува низ центарот на леќата. Ако точка p има координати $X = [X, Y, Z]^T$ релативни на координатен систем со центар во оптичкиот центар o , а z оската да е всушност оптичката оска, тогаш може да се види од слични триаголници на слика 11 дека координатите на p и нејзината проекција x се поврзани со така наречена *идеална перспективна проекција* [1]

$$x = -f \frac{X}{Z}, \quad y = -f \frac{Y}{Z} \quad (86)$$

Каде што f претставува *фокалното растојание*. Понекогаш проекцијата ја пишуваме како мапа π :

$$\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2; \quad X \mapsto x \quad (87)$$

Можеме да напишеме и $x = \pi(X)$. Треба да се напомене дека секоја друга точка која се наоѓа на линијата што минува низ p и o , исто така се проектира во истата точка x со координати $x = [x, y]^T$. Овој модел се нарекува *идеален модел на камера без леќа*. Овој модел е идеализација на бескрајно тенка леќа. Иако е можно да се направат уреди кои го користат моделот без леќа, овде тој се користи само како геометриски добра претпоставка за добро фокусиран систем за репрезентација на слика.



Слика 11. Модел на репрезентација на слика без леќа. Пресликувањето на точката p е во точката x во пресекот на зраците што минува низ оптичкиот центарот o и самата слика на растојание f позади оптичкиот центар.

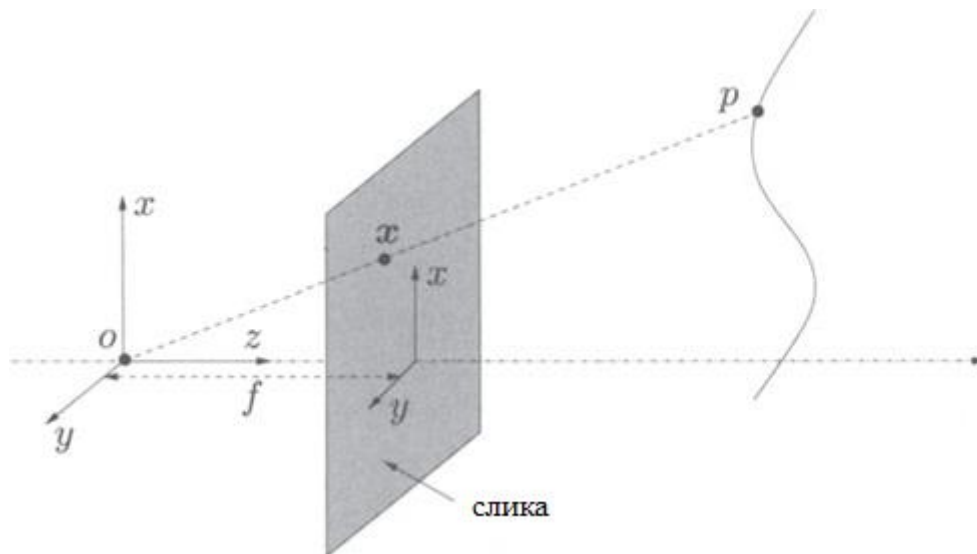
Негативниот знак од формулата (86) значи дека сликата што ја добиваме е свртена наопаку. За да го елиминираме овој ефект, можеме едноставно да ја завртиме сликата $(x, y) \mapsto (-x, -y)$, односно да ја ставиме пред оптичкиот центар $\{z = -f\}$ наместо зад него $\{z = +f\}$. Понатаму ќе го користиме овој начин на прикажување со проектирање на сликата пред оптичкиот центар, слика 12. Вака сликата $x = [x, y]^T$ од точката p е прикажана со:

$$x = f \frac{x}{z}, \quad y = f \frac{y}{z}. \quad (88)$$

Често го користиме знакот x да означиме хомоген приказ $[f \frac{x}{z}, f \frac{y}{z}, 1]^T \in \mathbb{R}^3$, се додека растојанието не е важно [1].

Во пракса големината на сликата е ограничена, па затоа не секоја точка p во просторот ќе генерира слика x во рамнината на сликата. Дефинираме видно поле (анг. field of view) кое претставува агол кој го зафаќа рамнината на сликата гледано од оптичкиот центар. Ако $2r$ е најголемото просторно проширување на сензорот (на пример должините на страните на CCD сензорот во камерата),

тогаш видното поле ќе биде $\theta = 2 \arctan\left(\frac{r}{f}\right)$. Доколку права рамнина се користи како рамнина за слика, аголот θ е секогаш помал од 180° .



Слика 12. Модел на репрезентација на слика без леќа пред оптичкиот центар. Пресликувањето на 3-Д точката p е точката x во пресекот на зраците кои што минуваат низ оптичкиот центар o и сликата на растојание f пред оптичкиот центар.

3.1.3 Геометриски модел за формирање на слика

Целта ни е да дознаеме која точка од реалниот простор одговара со која точка од сликата, со што директно би ја поврзале радијансата на реалната точка со интензитет или ирадијансата на точката на сликата [1]. За да воспоставиме прецизно совпаѓање помеѓу точките од 3-Д просторот (релативни на глобалниот координатен систем), со нивните проектирани слики во 2-Д просторот (релативни на координатниот систем на камерата), математичкиот модел мора да земе во предвид три типа на трансформации:

1. Трансформација на координати помеѓу координатниот систем на камерата и глобалниот координатен систем;
2. Проекција на 3-Д координати во 2-Д координати;
3. Трансформација на координати помеѓу можни избори на различни слики.

Во овој дел ќе објасниме упростен процес на формирање на слика, како серии на трансформации на координати. Инвертирајќи таков ланец на трансформации,

често се мисли за *калибрација на камера* (анг. Camera calibration), која што претставува клучен чекор за 3-Д реконструкција.

3.1.3.1 Идеална перспективна камера

Да разгледаме една точка p со координати $X_0 = [X_0, Y_0, Z_0]^T \in \mathbb{R}^3$ релативна на глобалниот координатен систем. Координатите $X = [X, Y, Z]^T$ од истата точка p релативна на координатен систем на камерата се претставени преку трансформацијата на круто тело, како $g = (R, T)$ од X_0 :

$$X = RX_0 + T \in \mathbb{R}^3 \quad (89)$$

Според моделот на камера без леќа кој го прикажавме претходно, точката X е проектирана на сликата во точка $[1]$:

$$x = \begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \end{bmatrix}. \quad (90)$$

Во хомогени координати оваа релација може да се запише како:

$$Z \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \quad (91)$$

Горната равенка може да ја запишеме и како:

$$Zx = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} X, \quad (92)$$

Каде што $X = [X, Y, Z, 1]^T$ и $x = [x, y, 1]^T$ сега се во хомогена репрезентација. Вредноста на Z која всушност е длабочината на точката p , односно нејзината одалеченост од камерата, обично ја пишуваме како позитивен скалар $\lambda \in \mathbb{R}_+$.

Треба да се напомене дека во равенката (92) може да направиме декомпозиција на матрицата во следниот облик $[1]$:

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (93)$$

При што ќе дефинираме две матрици:

$$K_f = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3 \times 3}, \quad \Pi_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \in \mathbb{R}^{3 \times 3} \quad (94)$$

Матрицата Π_0 се вика *стандардна* или *канонична проекциона матрица*. Од трансформацијата на координати имаме $X = [X, Y, Z, 1]^T$ [1],

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \quad (95)$$

И да сумираме, со користење на горната нотација, општиот геометриски модел на *идеална камера* може да се опише со:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix}, \quad (96)$$

или во друга форма:

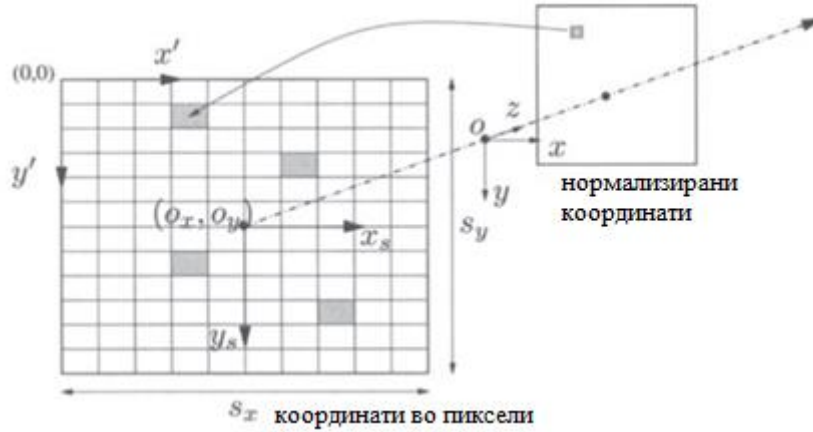
$$\lambda x = K_f \Pi_0 X = K_f \Pi_0 g X_0 \quad (97)$$

Ако фокалната должина f е позната, можеме да ја нормализираме на 1, со што овој модел се сведува на Еуклидова трансформација g , помножена со стандардната проекција Π_0 :

$$\lambda x = \Pi_0 X = \Pi_0 g X_0. \quad (98)$$

3.1.3.2 Камера со внатрешни параметри

Идеалниот модел на камера прикажан со равенка (97) е идеализиран, центриран во оптичкиот центар со едната оска што е порамнета со оптичката оска. Во пракса сликите што ги добиваме од дигиталната камера се претставени во форма на пиксели (i, j) , со координатен почеток не во центарот на сликата туку во горниот лев агол. За да можеме да ја искористиме равенката (97) во реално сценарио мора да воспоставиме однос помеѓу низата од пиксели која ја добиваме и моделот за идеална камера [1].



Слика 13. Трансформација од нормализирани координати во координати во пиксели.

Прво треба да одредиме кои мерни единици ќе ги користиме за x и y оската. Ако (x, y) се означени во милиметри, а (x_s, y_s) се скалирана верзија што одговараат на координати од пиксели, тогаш трансформацијата може да се опише како матрица на скалирање

$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (99)$$

што зависи од големината на пикселот (во милиметри) долж x и y слика 13. Кога $s_x = s_y$ тогаш пикселот е во форма на квадрат. Во реалноста овие вредности ретко се еднакви, па пикселот е правоаголник. Овде x_s и y_s се наведени за точката на пресек (местото каде z - оската ја сечи сликата), а каде што пикселот (i, j) е означен со позитивни вредности релативно на горниот лев агол од сликата. Затоа мораме да ја преместиме точката, односно центарот на координатниот систем во горниот лев агол од сликата како што е прикажано на слика 13,

$$x' = x_s + o_x, \quad (100)$$

$$y' = y_s + o_y, \quad (101)$$

Каде што (o_x, o_y) се координатите на точката на пресек, изразени во пиксели релативно на координатниот систем на сликата. Па точните координати на сликата се дадени со векторот $\mathbf{x}' = [x', y', 1]^T$ наместо идеалните координати на

сликата $x = [x, y, 1]^T$. Чекорите на трансформација на координати може да бидат запишани во хомоген приказ

$$x' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (102)$$

Каде што x' и y' се реалните координати во пиксели. Во случај кога пикселите не се правоаголни ја имаме следната матрица за скалирање:

$$\begin{bmatrix} s_x & s_\theta \\ 0 & s_y \end{bmatrix} \in \mathbb{R}^{2 \times 2}, \quad (103)$$

каде што s_θ се нарекува фактор на закосеност и θ е аголот помеѓу двете оски x и y . Матрицата на трансформација од равенката (102) ја има генералната форма:

$$K_s = \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3 \times 3}. \quad (104)$$

Во многу практични апликации се апроксимира $s_\theta = 0$. Ако ги комбинираме проекцискиот модел со скалирањето и транслацијата ќе добиеме пореалистичен модел од трансформацијата помеѓу хомогени координати на 3Д точката релативно на координатниот систем од камерата и хомогени координати на пикселите од сликата.

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \quad (105)$$

Од равенката се гледа дека ефектот на реалната камера е изразен преку две фази [1]:

- Првата фаза е стандардна перспективна проекција во однос на нормализираниот координатен систем (со фокална должина од 1). Ова е окарактеризирано со стандардната проекциска матрица $\Pi_0 = [I, 0]$.
- Втората фаза е дополнителна трансформација (на веќе добиената слика x) која зависи од параметрите на самата камера, како што се фокалната должина f , факторите на скалирање s_x, s_y и s_θ , и координатите на изместениот центар o_x, o_y .

Втората трансформација се карактеризира со комбинација на две матрици K_s и K_f :

$$K = K_s K_f = \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \quad (106)$$

Со спојување на K_s и K_f проекциската равенка може да ја напишеме во облик:

$$\lambda x' = K \Pi_0 X = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (107)$$

Матрицата $\Pi_0(3 \times 4)$ е матрица на перспективната проекција. Горно триаголната матрица $K(3 \times 3)$ ги содржи сите внатрешни параметри за дадената камера, па според тоа и се нарекува *матрица за внатрешни параметри* или **матрица на калибрација** на камерата [1]. Елементите од матрицата K ја имаат следната геометричка интерпретација:

- o_x : x – координата на оптичкиот центар во пиксели,
- o_y : y – координата на оптичкиот центар во пиксели,
- $fs_x = \alpha_x$: големина во единица должина на пиксел по хоризонтала,
- $fs_y = \alpha_y$: големина во единица должина на пиксел по вертикала,
- α_x/α_y : однос на должини σ ,
- fs_θ : закосеност на пикселот, најчесто е блиску до нула.

Висината на пикселот не мора да е еднаква со ширината, освен ако односот на должините е 1. Кога матрицата на калибрација е позната, калибрираните координати x можат да се најдат од координатите во пиксели x' со инверзија на K на следниот начин:

$$\lambda x = \lambda K^{-1} x' = \Pi_0 X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (108)$$

Информациите за матрицата K може да се добијат преку процес наречен калибрација на камера. Да сумираме, геометрискиот однос помеѓу точка со координати $X_0 = [X_0, Y_0, Z_0, 1]^T$ релативно на глобалниот светски координатен систем и соодветната точка на сликата што сме ја добиле од камерата со

координати $x' = [x', y', 1]^T$ (во пиксели) зависи од движењето на крутото тело опишано со (R, T) помеѓу глобалниот светскиот координатен систем и координатниот систем врзан за камерата (кој понекогаш се нарекува **калибрација на надворешни параметри**), од идеалната проекциска матрица Π_0 , и од внатрешните параметри на камерата K . Целосниот модел за формирањето на сликата е опишан со следната равенка:

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = K \Pi_0 X = \begin{bmatrix} f s_x & f s_\theta & o_x \\ 0 & f s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \quad (109)$$

Или во форма на матрица:

$$\lambda x' = K \Pi_0 X = K \Pi_0 g X_0 \quad (110)$$

$$\text{или } \lambda x' = K \Pi_0 X = [KR, KT] X_0. \quad (111)$$

Матрицата $K \Pi_0 g = [KR, KT]$ која е (3×4) матрица, се вика општа проекциска матрица Π , за да се разликува од стандардната проекциска матрица Π_0 .

Според тоа равенката (111) се запишува како:

$$\lambda x' = \Pi X_0 = K \Pi_0 g X_0. \quad (112)$$

Споредено со идеалниот модел на камера од равенката (98), се што сменивме е стандардната проекциска матрица Π_0 со општата проекциска матрица Π .

На крај за да ги добиеме координатите на сликата (x', y', z') и да ја видиме нелинеарната природа на равенката за перспективна проекција, ќе ја поделиме равенката (112) со факторот за скалирање λ :

$$x' = \frac{\pi_1^T X_0}{\pi_3^T X_0}, \quad y' = \frac{\pi_2^T X_0}{\pi_3^T X_0}, \quad z' = 1, \quad (113)$$

Каде што $\pi_1^T, \pi_2^T, \pi_3^T \in \mathbb{R}^4$ се трите реда од проекциската матрица Π [1].

3.1.3.3 Радијална дисторзија

Како продолжување на линеарна дисторзија објаснета од параметрите во K , ако користиме камера со широко видно поле, честопати имаме голема

дисторзија и по радијалните насоки. Наједноставниот модел за таква дисторзија е:

$$x = x_d(1 + a_1r^2 + a_2r^4), \quad (114)$$

$$y = y_d(1 + a_1r^2 + a_2r^4), \quad (115)$$

Каде (x_d, y_d) се координати од дисторзирани точки, $r^2 = x_d^2 + y_d^2$, а a_1, a_2 се дополнителни параметри на камерата што ја одредуваат величината на дисторзија. Неколку алгоритми и софтверски пакети се достапни за отстранување на радијална дисторзија. Најчесто употребуван пристап е преку калибрација на камера. Ако овој пристап не е можен, параметрите на радијалната дисторзија може да се одредат директно од сликите. Едноставен метод на радијална дисторзија :

$$x = c + f(r)(x_d - c), \quad (116)$$

$$f(r) = 1 + a_1r + a_2r^2 + a_3r^3 + a_4r^4, \quad (117)$$

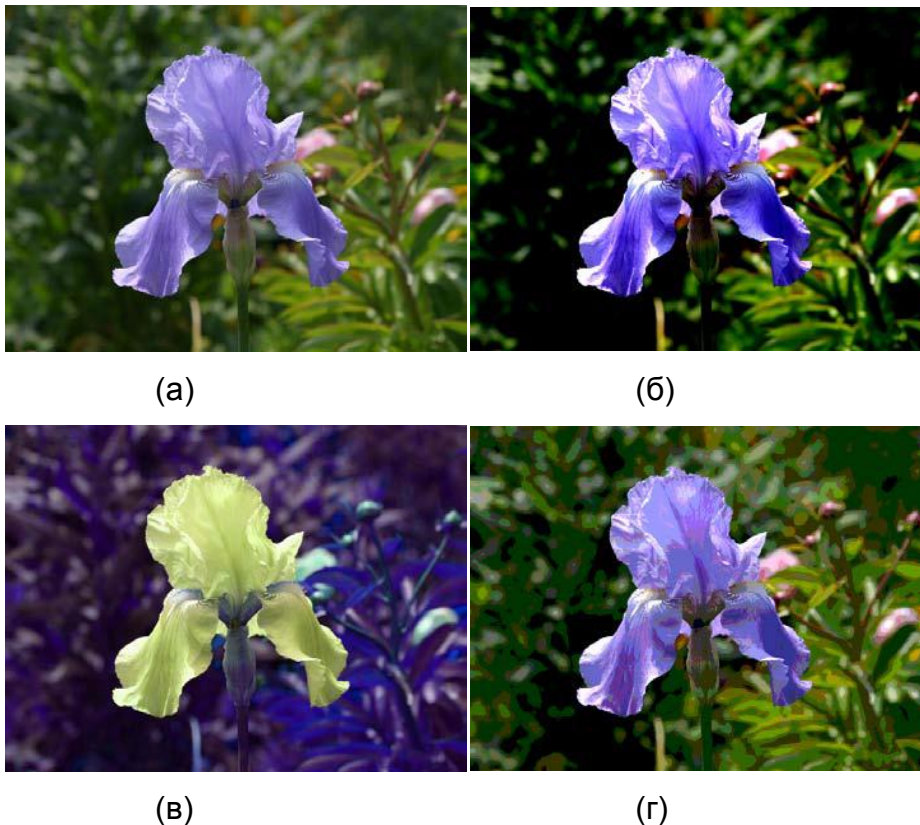
Каде $x_d = [x_d, y_d]^T$ се дисторзираните координати на сликата, а $r^2 = \|x_d - c\|^2$, $c = [c_x, c_y]^T$ е центарот на дисторзијата, но не мора да се совпаѓа со центарот на сликата, а $f(r)$ е фактор на корекција на дисторзијата. Методот е собирање множество од прави линии од околината и ги пресметува најдобрите параметри за модел на радијалната дисторзија, кој што ќе ги претвори искривените линии од сликите во прави сегменти. Овој модел се користи за да трансформира слика 14 лево во 14 десно. Затоа понатаму во текстот ќе сметаме дека радијалната дисторзија е отстранета, а камера ќе биде опишана само со параметарската матрица K .



Слика 14. Лево е слика направена со камера со кратко фокално растојание, каде што правите линии во сцената се искривуваат. Десно е слика со отстранета радијална дисторзија.

3.2 Процесирање на слика

Сега откако видовме како се формираат сликите, преку интеракција на 3-Д елементите од сцената, осветлувањето, сензорите на камерата и леќите, да ја погледнеме првата фаза која се одвива во повеќето апликации на компјутерска визија. Имено тоа е употребата на процесот за обработка на слика кој служи за подготовка на сликата и нејзина конверзија во потребната форма за понатамошна анализа. Примери за таквиот процес вклучуваат корекции на изложеноста и балансирање на бои, намалување на шумови на слика, зголемување на острина или израмнување на сликата преку ротација како на слика 15. Повеќето апликации на компјутерска визија, како што е компјутерското фотографирање па дури и препознавање, бараат внимание во дизајнирањето на фазите во процесот на обработка на слики со цел да се постигнат прифатливи резултати [3].





(д)

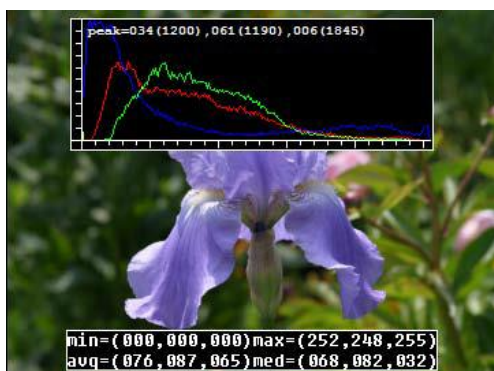


(f)

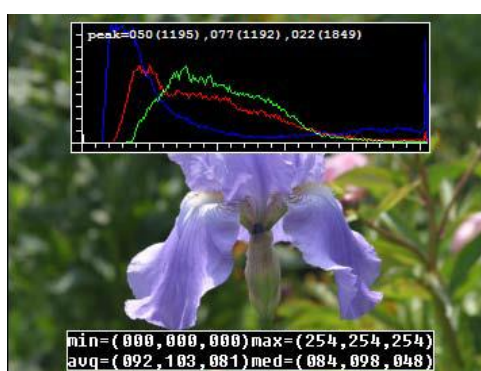
Слика 15. Некои операции за процесирање на слики: (а) оригинална слика; (б) зголемен контраст; (в) промена на боја; (г) posterized; (д) заматена; (f) ротирана.

3.2.1 Точкасти оператори

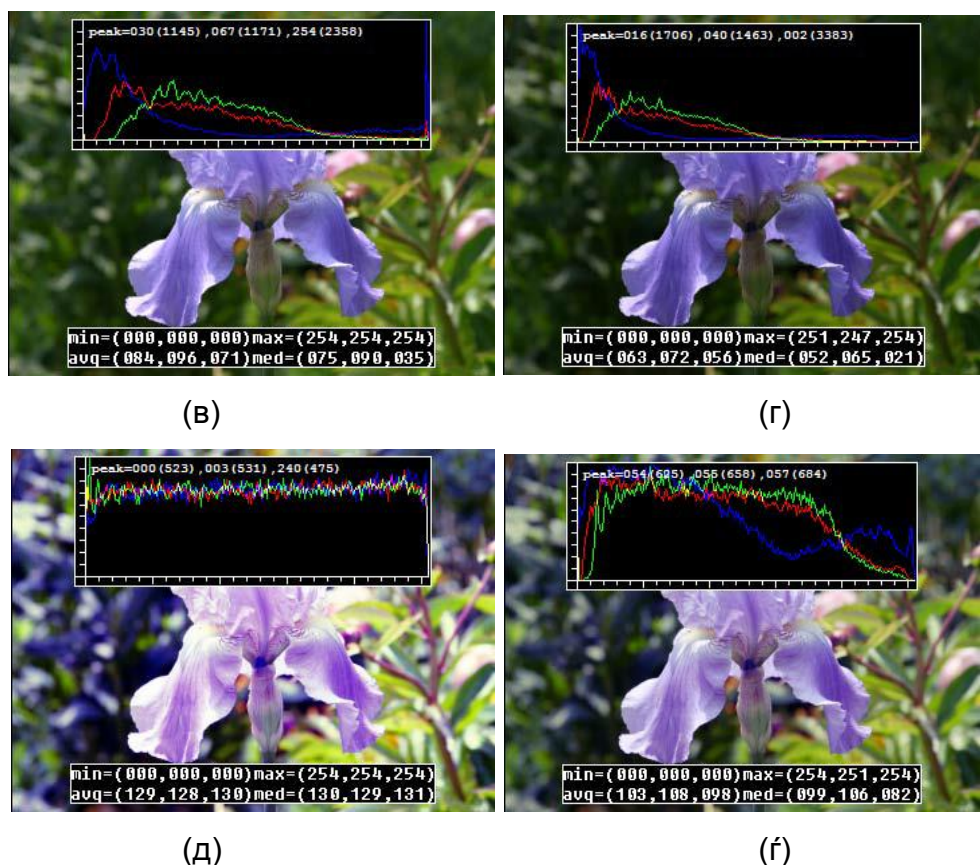
Наједноставните видови на процесирање на слика се *точкастите оператори*, каде што секоја излезна вредност на пиксел зависи само од коресподентната вредност на влезниот пиксел. Точкастите оператори вклучуваат подесувања на светлина и контраст, слика 16, како и корекција на боја и трансформации. Во литературата за процесирање на слики овие оператори се познати како *точкасти процеси*.



(a)

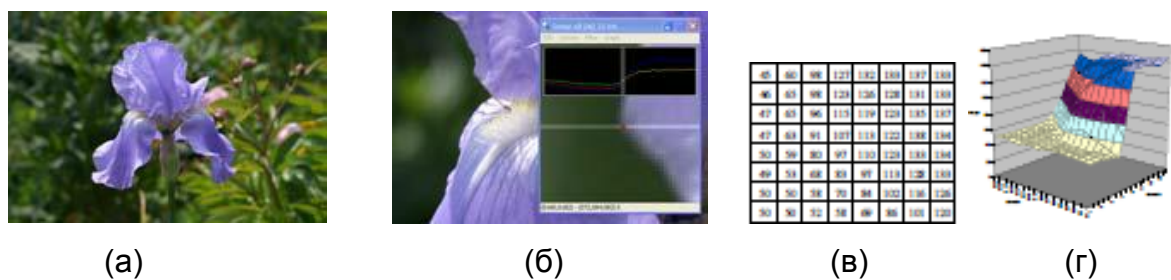


(б)



Слика 16. Некои операции за процесирање на слика: (а) оригинална слика со дијаграм на три бои; (б) зголемено осветлување; (в) зголемен контраст; (г) зголемена гама; (д) целосно израмнување на дијаграм; (е) делумно израмнување на дијаграм

Ќе почнеме со преглед на едноставни точкасти оператори како што е скалирање на светлина и додавање на слика. Ќе разгледаме како може да се манипулира со боите во сликите. Ќе ги претставуваме *исцртување на слика* и *операции за подлога на слика*, кои што играат важна улога во компјутерското фотографирање и апликациите на компјутерска графика [3]. За крај ќе го објасниме општиот процес на изедначување на дијаграм.



Слика 17. Визуализација на податоците на слика: (а) оригинал слика; (б) исечок од слика; (в) мрежа со бројки; (г) дводимензионална површина. Сликите (в) и (г), беа прво конвертирани во нијанси на сиво.

3.2.1.1 Трансформација на пиксел

Општа операција за процесирање на слика претставува функција која што за влез има една или повеќе слики, а произведува една излезна слика. Ова може да се претстави со формула како

$$g(x) = h(f(x)) \text{ или } g(x) = h(f_0(x), \dots, f_n(x)), \quad (118)$$

каде што x е во D -димензионален домен на функциите (обично за слики $D=2$), а функциите f и g работат на опсегот, кој што може да е скалар или вектор т.е. за слики во боја или 2-Д движење. Кај примерна слика, доменот се состои од точен број на локации на пиксели, $x = (i, j)$, па може да запишеме

$$g(i, j) = h(f(i, j)). \quad (119)$$

Слика 17 покажува дека слика може да биде прикажана како слика во боја, како мрежа со броеви или како дводимензионална функција [3].

Два често употребувани точкасти процеси се множење и собирање со константа,

$$g(x) = af(x) + b. \quad (120)$$

Параметрите $a > 0$ и b се нарекуваат засилувачки и грешка параметри, респективно; понекогаш овие параметри го контролираат контрастот и светлината, прикажано на слика 16 б-в. Овие параметри можат просторно да варираат

$$g(x) = a(x)f(x) + b(x), \quad (121)$$

Всушност стимулирајќи го степенеститот филтер за густина кој се користи од фотографите за селективно да го затемнат небото, или да го подесат сенчањето во оптичкиот систем.

Повеќекратна добивка (општо или просторно варирајќи) е линеарен оператор, бидејќи го подржува принципот на суперпозиција,

$$h(f_0 + f_1) = h(f_0) + h(f_1). \quad (122)$$

Операторите како квадрирање на слика не се линеарни.

Друг често употребуван оператор со два влезе е линеарниот мешан оператор,

$$g(x) = (1 - \alpha)f_0(x) + \alpha f_1(x). \quad (123)$$

Со варирање на α од $0 \mapsto 1$, овој оператор може да се користи за да се изведи вкрстено решавање помеѓу две слики или видеа, како што се прикажува во филмската продукција, или како компонента на алгоритмите за преливање (промена) на слики од една во друга [3].

Една многу употребувана нелинеарна трансформација што е користи врз сликите пред понатошно процесирање е гама корекцијата. Оваа корекција се користи за да се отстрани нелинеарното мапирање помеѓу влезната радијанса и вредноста на пикселите. За да се инвертира гама мапирањето кое што е добиено од сензорот, може да користиме

$$g(x) = [f(x)]^{1/\gamma}, \quad (124)$$

Каде вредноста на гама $\gamma = 2.2$, е разумно избрана за да одговара во повеќето дигитални камери.

3.2.1.2 Трансформации на боја

Сликите во боја може да се третираат како произволни функции од векторски-вредности, или колекција од независни групи, па има смисла да се мисли за нив како сигнали со големо значење за поврзување со процесот на формирање на слика, дизајн на сензорот и човековата перцепција [3].

Да земеме на пример, при осветлување на слика трите канали со боја да имаат константни вредности, како на слика 16 б.

Всушност, додавајќи ја истата вредност на секој канал со боја, не само што го зголемува интензитетот на секој пиксел, туку исто така влијае на нијансите и заситеноста на пикселите. Како може да дефинираме и манипулираме со такви квантитети со цел да ги постигнеме посакуваните перцептивни ефекти?



(a)

(б)

(в)

(г)

Слика 18 Подлога и исцртување на слика: (а) основна слика; (б) отстранета позадина; (в) приказ во нијанса на сиво; (г) ново исцртување.

Последователните координати или едноставни соодноси со боја, може првин да се пресметаат, а потоа по манипулирањето (осветлување) со светлината Y одново да пресметаат валидна RGB слика со истите нијанси и густина.

Балансирање на боја (т.е. надополнување за блескавото осветлување) може да биде изведено со множење на секој канал со различен фактор, или со посложениот процес на мапирање на XYZ просторот, променувајќи ја номиналната бела точка и мапирајќи повторно назад во RGB. Сето тоа може да биде запишано како линеарна 3×3 матрица на трансформација [3].

3.2.1.3 Составување и поставување подлога

Во повеќето апликации со визуелни ефекти и корекција на слика, често се користи сечење на преден објект од една сцена и истиот да се постави врз друга позадина, како на слика 18. Процесот на отстранување на објектот од оригиналната слика се нарекува подметка (анг. *matting*), а процесот на внесување во друга слика се нарекува составување (анг. *compositing*).

Приказот на исечениот објект од сликата, меѓу двата процеса се нарекува алфа подесена слика во боја (анг. *alpha-matted color image*), слика 18 б-в. Како дополнување на трите канали со боја RGB, алфа подесената слика содржи четврти алфа канал α (или A) што го опишува количеството на непроѕирност или делумна покриеност на секој пиксел, слика 18в и 19б. Непроѕирноста е спротивна од проѕирноста (транспарентноста). Пикселите во објектот се целосно непроѕирни ($\alpha = 1$), додека пикселите што се надвор од објектот се проѕирни ($\alpha = 0$). Пикселите кои се на границата на објектот глатко варираат помеѓу двете екстрими, со што се сокриваат видливите запци што се појавуваат доколку се користат бинарни затемнувања [3].

$$\begin{array}{ccccccc}
 \begin{array}{|c|c|c|} \hline \text{Green} & \text{Blue} & \\ \hline \text{Blue} & \text{Green} & \\ \hline \end{array} & \times (1 - & \begin{array}{|c|c|c|} \hline \text{Dark} & \text{Light} & \\ \hline \text{Light} & \text{Dark} & \\ \hline \end{array} &) + & \begin{array}{|c|c|c|} \hline \text{Dark} & \text{Dark} & \\ \hline \text{Dark} & \text{Dark} & \\ \hline \end{array} & = & \begin{array}{|c|c|c|} \hline \text{Dark} & \text{Dark} & \\ \hline \text{Dark} & \text{Dark} & \\ \hline \end{array} \\
 B & & \alpha & & \alpha F & & C \\
 \text{(а)} & & \text{(б)} & & \text{(в)} & & \text{(г)}
 \end{array}$$

Слика 19. Композициона равенка $C = (1 - \alpha)B + \alpha F$. Сликите се сликани од близина на регион на влакно во горниот десен дел од лавот на слика 18.

За да се композира нова слика (исечен објект) врз стара слика (друга позадина), операторот крај се користи:

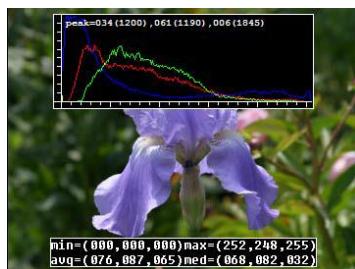
$$C = (1 - \alpha)B + \alpha F. \quad (125)$$

Овој оператор го намалува влијанието на сликата со позадина B со фактор $(1 - \alpha)$, а потоа ги додава вредностите за бојата (и непроѕирноста) соодветно на исечениот објект F , како на слика 19.

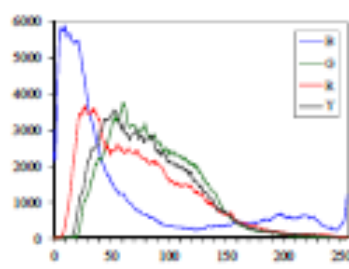
Во многу ситуации, полесно е да се претстават боите од предниот исечен објект во форма пред множењето, т.е. да се зачуваат вредностите на αF . Репрезентација на пред множење на RGBA е пожелна за неколку причини, вклучувајќи ја можноста за заматување или ротирање на *alpha-matted* сликите без дополнителни компликации. Сепак за подесување се употребува локална постојана боја, се користат и чистите не множени бои на исечениот објект, бидејќи остануваат константи или многу бавно се променуваат во близината на работ на објектот [3].

3.2.1.4 Изедначување на хистограм

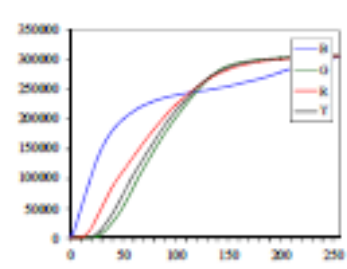
Поради тоа што осветлувањето и добивањето на контрола можат да го подобрат изгледот на една слика, како може да ги одбереме нивните најдобри вредности? Еден пристап е да се погледнат вредностите на најтемните и најсветлите пиксели во една слика и да ги мапираме со чиста црна и чиста бела. Друг пристап може да биде да се најде средна вредност во сликата, да се постави кон средна сива боја и да го прошириме опсегот така да попрецизно да ги исполни вредностите.



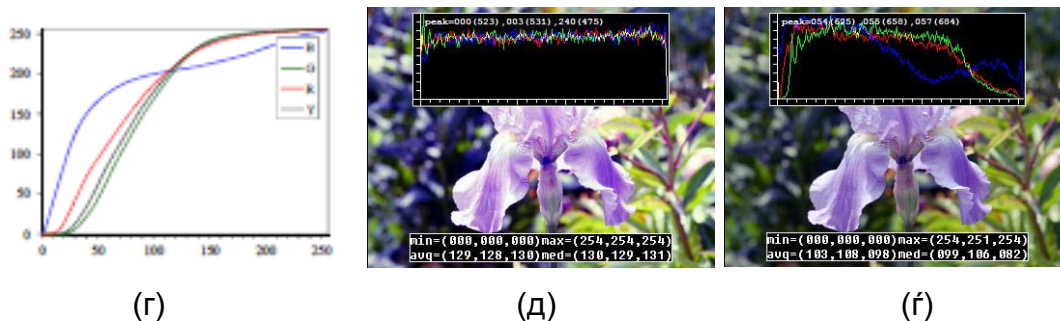
(a)



(б)



(в)



Слика 20. Анализа на хистограм и споредба.

Како може да визуелизираме група на светлостни вредности во слика со цел да се тестира одредена евристика? Одговорот е да се претстави на хистограм со вредностите од секој канал на боја и осветленоста како на слика 20б. Од тука можеме да добиеме релевантна статистика за минимални, максимални и средни вредности на интензитетот. Да се забележи дека сликата на слика 20а има и темни и светли вредности, но средните вредности ги има многу послабо [3]. Со цел да се потемнат некои од светлите вредности и да се осветлат некои од темните вредности, а сеуште целосно да се користи достапниот динамички опсег, потребно е да се изведе изедначување на хистограм (histogram equalization). Односно тоа значи да се најде функција за мапирање $f(I)$ поради која што дијаграмот е израмнет. Трикот за пронаоѓање на такво мапирање е истиот што луѓето го користат за производство на случајни примероци од веројатносна функција за густина, што е всушност прво да се пресмета кумулативната функција на дистрибуција, слика 20в. Да го претпоставиме оригиналниот приказ на дијаграмот $h(I)$ како дистрибуција на оценки во клас после спроведен испит. Како може да ја мапираме одредена оценка соодветно на нејзините проценти, така да студентите со резултат 75% , биле подобри од $\frac{3}{4}$ од нивните колеги? Одговорот е да се интегрира дистрибуцијата $h(I)$ за да се добие кумулативна дистрибуција $c(I)$,

$$c(I) = \frac{1}{N} \sum_{i=0}^I h(i) = c(I-1) + \frac{1}{N} h(I), \quad (126)$$

Каде што N е бројот на пиксели во сликата или студентите во класот. За секоја дадена оценка или интензитет, може да погледнеме соодветен процент $c(I)$ и да ја одредиме конечната вредност што пикселот треба да ја има. Кога работиме со осум битни вредности на пиксел, I и c оските се скалирани од $[0,255]$. За ова повеќе во [3].



Слика 21. Локално адаптивно споредување на дијаграм: (а)оригинал слика; (б) блокови на споредба; (в) целосна адаптивна споредба.

Слика 20г го покажува резултатот од аплицирањето $f(I) = c(I)$ во оригиналната слика. Како што може да видиме резултирачкиот хистограм е израмнет; со тоа и добиената слика (израмнета е во смисла дека има контраст и не е заматена). Еден начин да се компензира за ова е делумно да се компензира за нерамнините на хистограмот, т.е. употребувајќи функција за мапирање $f(I) = \alpha c(I) + (1 - \alpha)I$, која што е линеарно вклопување помеѓу кумулативната дисторзирана функција и трансформацијата на идентитет (права линија). Како што се гледа на слика 20д, добиената слика задржила повеќе од оригиналната дисторзија според нијанси на сиво, а има попривлечен баланс. Друг потенцијален проблем со израмнување на хистограм, (или општо со осветлување на слика), е тоа дека шумот во темни региони може да се зголеми и да стане повеќе видлив.

3.2.2 Линеарно филтрирање

Овде ќе погледнеме оператори за линеарно филтрирање, кои што вклучуваат комбинирање на соседни пиксели во мали групи. Најчесто користен оператор за групирање на соседни пиксели е линеарното филтрирање, во кое што вредноста на излезниот пиксел е добиена како збир од вредностите на влезниот пиксел, слика 22,

$$g(i, j) = \sum_{k, l} f(i + k, l) h(k, l). \quad (127)$$

Ентитетите во $h(k, l)$ се нарекуваат коефициенти на филтрирање. Операторот може да биде поедноставно претставен како

$$g = f \otimes h. \quad (128)$$

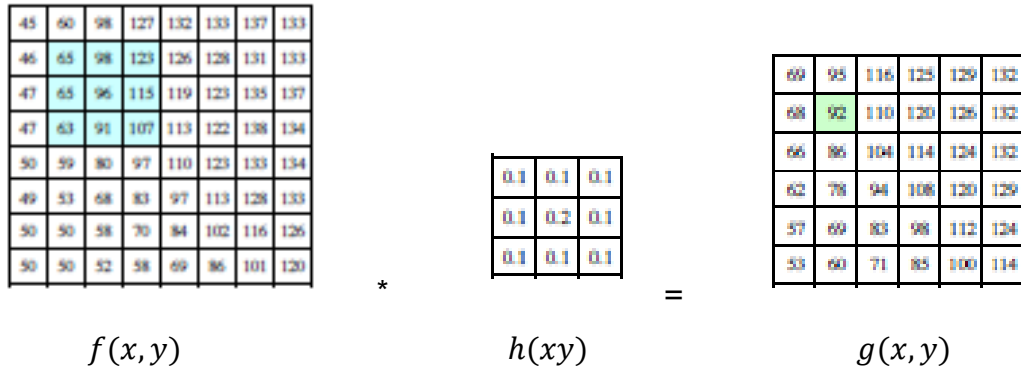
Друга варијанта на следнава формула е :

$$g(i, j) = \sum_{k, l} f(i - k, j - l) h(k, l) = \sum_{k, l} f(k, l) h(i - k, j - l), \quad (129)$$

Каде знакот за поместување во f е обратен. Ова се нарекува оператор за *присоединување*

$$g = f * h \quad (130)$$

а h се нарекува функција на импулсна реакција.



Слика 22. Соседно филтрирање: сликата од лево е присоединета со филтерот во средина за да се добие сликата од десно. Светло сините пиксели го означуваат соседството од кое е добиен светло зелениот пиксел.

Причината за ова име е поради тоа што основната функција, h , кога е присоединета со импулсен сигнал $\delta(i, j)$ (слика која што има вредност 0 насекаде освен во центарот), повторно се добива истата $h * \delta = h$, каде што корелацијата го создава рефлектирачкиот сигнал.

Равенката (129) може да биде образложена како производ на поместена функција на импулсна реакција $h(i - k, j - l)$ помножена со вредностите на влезните пиксели $f(k, l)$. Кај присоединувањето важи и комутативното и асоцијативното својство.

Корелацијата и присоединувањето се линеарни непроменливо поместувачки оператори (анг. linear shift invariant LSI), кои што го подржуваат и принципот на множење

$$h \circ (f_0 + f_1) = h \circ f_0 + h \circ f_1, \quad (131)$$

и принципот на непроменливо поместување

$$g(i, j) = f(i + k, j + l) \Leftrightarrow (h \circ g)(i, j) = (h \circ f)(i + k, j + l), \quad (132)$$

Кое што означува дека поместувањето на сигнал одговара со аплицирање на операторот (\circ означува LSI (линеарно непроменливо поместување) оператор. Друга замисла на непроменливото поместување е дека операторот се однесува еднакво насекаде.



(a)



(б)



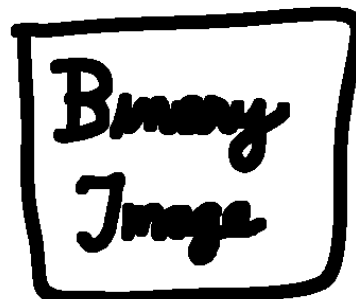
(в)



(г)



(д)



(f)



(e)



(ж)

Слика 23. Некои соседни операции: (а) оригинална слика; (б) заматена; (в) изострена; (г) израмнета со филтер за задржување на раб; (д) бинарна слика; (ѓ) раширена; (е) трансформација на дистанца; (ж) конектирани компоненти. За раширените и конектираните компоненти, црните пиксели се претпоставува дека се активни, т.е. дека имаат вредност 1.

$$\begin{bmatrix} 72 & 88 & 62 & 52 & 37 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \end{bmatrix} \Leftrightarrow \frac{1}{4} \begin{bmatrix} 2 & 1 & . & . & . \\ 1 & 2 & 1 & . & . \\ . & 1 & 2 & 1 & . \\ . & . & 1 & 2 & 1 \\ . & . & . & 1 & 2 \end{bmatrix} \begin{bmatrix} 72 \\ 88 \\ 62 \\ 52 \\ 37 \end{bmatrix}$$

Слика 24. Едно димензионален сигнал за присоединување како множење на матрични вектори, $g = Hf$.

Повремено поместувачка верзија на корелација или присоединување може да се користи како

$$g(i, j) = \sum_{k,l} f(i - k, j - l) h(k, l; i, j), \quad (133)$$

Каде што $h(k, l; i, j)$ е основата за присоединување на пиксел (i, j) . На пример таква просторно варирачка основа може да се употреби за да се моделира заматена слика поради варирањето по длабочина на фокусот.

Корелацијата и присоединувањето можат да се запишат како производ од множење на матрица и вектор, ако прво ги конвертираме дводимензионалните слики $f(i, j)$ и $g(i, j)$ во растер – подредени вектори f и g ,

$$g = Hf, \quad (134)$$

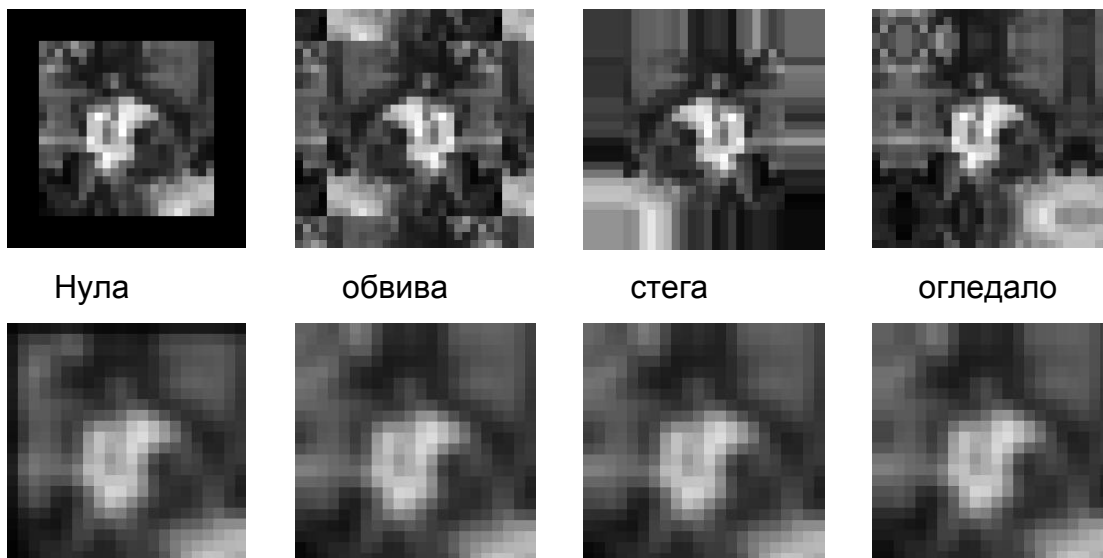
Каде што H матрицата ја содржи основата за присоединување. Слика 24 покажува како едnodимензионално присоединување може да се претстави во вектор-матрица форма.

❖ Пополнување (оформување)

Да се забележи дека производот матрици на слика 24 страда од ефекти на рамка, т.е. резултатите од филтрирање на слика од оваа форма ќе доведат до затемнување на пикселите на краевите од сликата. Ова се случува поради тоа што за ефективно пополнување на оригиналната слика се користи вредност 0, секаде каде што присоединувачката основа е проширена преку границата на рамката на оригиналната слика.

За да се компензира ова, постојат бројни алтернативни пополнувања (оформувања) или проширени модови:

- Нула: сите пиксели што се надвор од оригиналната слика да се постават на нула;
- Константа (боја на рамка): сите пиксели надвор од оригиналната слика да се постават на специфична вредност која е одбрана за рамката;
- Стега (реплицира или стега до раб): да се повторуват пикселите на рабовите бесконечно;
- кружно обвивање (повторувај): кружи околу сликата во кружна конфигурација;
- огледало: рефлектира пиксели по рабовите на сликата;
- проширена: проширување на сигналот преку намалување на огледалната верзија на сигналот од вредноста на пикселот што се наоѓа на работ.



Заматена нула норм. нула заматена стега заматено огледало

Слика 25. Пополнување на рамка (горниот ред) и резултати од заматување на пополнуваната слика (долниот ред). Сликата нормализирана на нула е резултат од делење на заматената пополнета со нула RGB слика со нејзината коресподентна алфа вредност.

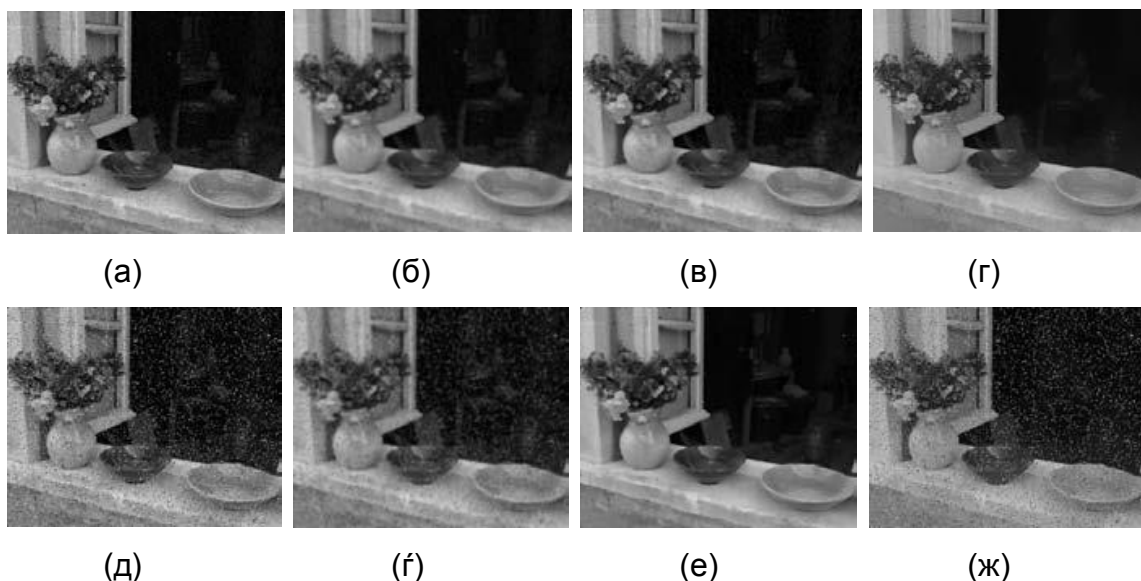
Сликата 25 го покажува ефектот на пополнување на слика со секој од горе наведените механизми, а потоа заматување на добиената слика. Како што може да се види кај сликите каде има нулта вредност за пополнување, рабовите потемнуваат, кај реплицираното пополнување вредностите на рабовите се

навнатре поставени , огледално пополнување ги зачувува бои близу рамката. Проширено пополнување ги задржува пикселите фиксни. Алтернатива на пополнувањето е заматување на нултата RGBA слика, а потоа да се подели добиената слика со нејзината алфа вредност за да се отстрани ефектот на затемнување [3]. Резултатите можат да бидат доста добри како што е прикажано на нулта нормализираната слика на слика 25.

3.2.3 Нелинеарно филтрирање

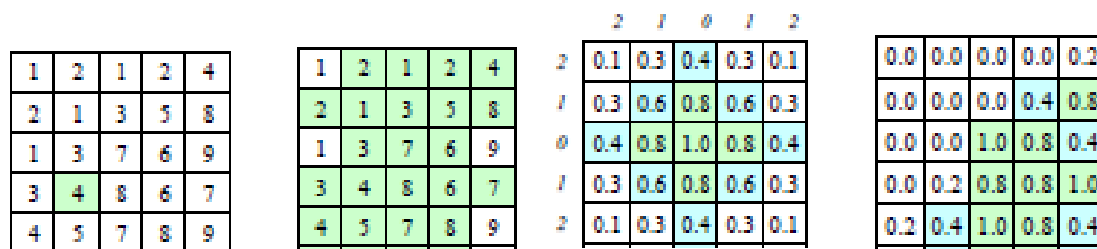
Како што видовме со линеарното филтрирање може да се изведуваат широк опсег на трансформации на слика. Со нелинеарното филтрирање пак, како што се медијана или билатерално филтрирање, може да се постигнат уште подобри резултати.

При линеарно филтрирање, реакцијата кон збир на два сигнала и иста како и кон збир на индивидуални реакции. Ова е исто како и за тоа дека секој излезен пиксел е збир од неколку влезни пиксели. Линеарните филтри се полесни за составување и се подложни на фреквентна анализа. Во повеќето случаи подобра изведба може да се добие со употреба на нелинеарни комбинации на соседни пиксели. Да ја погледнеме слика 26д, каде што шумот, наместо Гаусов, тој претставува ударен шум, т.е. повремено има големи вредности. Во овој случај обичното замаглување со Гаусовиот филтер, не успева да ги отстрани и притоа замени пикселите со шум со слаби, но сепак видливи точки, слика 26г.



Слика 26. Медијана и билатерално филтрирање: (а) оригинална слика со гаусов шум; (б) гаусово филтрирање; (в) медијана филтрирање; (г)

билатерално филтрирање; (д) оригинална слика со ударен шум; (ф) гаусово филтрирање; (е) медијана филтрирање; (ж) билатерално филтрирање;



(а) медијана=4 (б) α -средна=4.6 (в) обласен филтер (г) опсег филтер

Слика 27. Медијана и билатерално филтрирање: (а) среден пиксел (зелен); (б) селектирани пиксели; (в) обласен филтер (вредностите долж рабовите се растојанието на пикселите); (г) опсег филтер.

❖ Медијана филтрирање

Подобар филтер за употреба во овој случај е медијана филтрирање, кое што ги селектира средните вредностите од секоја група на соседни пиксели, слика 27а. Бидејќи вредноста на ударниот шум обично е надвор од вистинските вредности на групата соседни пиксели, медијана филтрирањето е способно да ги исфилтрира лошите пиксели, слика 26в.

Една негативност на медијана филтрирањето е тоа што одбира само една вредност од влезен пиксел за да ја замени вредноста на излезен пиксел, не е нешто повеќе ефикасен од просечното одстранувањето на регуларен гаусов шум. Подобар избор можеби да е α - исечена средина, која што ги подредува сите пиксели, освен факториелот α , дека се најмали и најголеми, слика 27б.

Друга можност е да се пресмета тежинската медијана, во која секој пиксел се употребува одреден број на пати, во зависност од неговото расојание од центарот. Ова излегува да е еквивалентно со минимизирањето на тежинската објективна функција :

$$\sum_{k,l} \omega(k,l) |f(i+k, j+l) - g(i,j)|^p, \quad (135)$$

Каде што $g(i,j)$ е посакуваната излезна вредност, а $p = 1$ за тежинската медијана. Вредноста $p = 2$ е вообичаената тежинска медијана, која што е еквивалентна во корелацијата со формулата (127), после нормализирање со збирот на тежините.

Нелинеарното измазнување има друго својство, кое е особено важно бидејќи ударниот шум е доста ретко присутен кај денешните камери. Таквото филтрирање уште повеќе ја зачувува рамката, т.е. има помала тенденција да ги ублажи рабовите при филтрирањето на високо фреквентни шумови.

Да ја земеме во предвид сликата со шум 26а. Со цел да се отстрани повеќето од шумот, гаусовото филтрирање е принудено да ги ублажи високо фреквентните детали, кое што посебно се забележливо близу јаки кошиња. Медијана филтрирањето работи подобро, но како што споменавме претходно, не работи добро со измазнување после прекини со работата.

Можеме да се обидиме да го користиме α – исечена средина или тежинска медијана, но овие техники сеуште имаат тенденција да ги заоблат острите кошиња, бидејќи важноста на пикселите во измазнетата зона доаѓа од распределбата на позадината.

❖ Билатерално филтрирање

Во билатералното филтрирање вредноста на излезниот пиксел зависи од тежинската комбинација на вредностите на групираниите соседни пиксели

$$g(i, j) = \frac{\sum_{k,l} f(k, l) \omega(i, j, k, l)}{\sum_{k,l} \omega(i, j, k, l)}. \quad (136)$$

Тежинските коефициенти $\omega(i, j, k, l)$ зависат од производот од обласната средина слика 27в,

$$d(i, j, k, l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2}\right), \quad (137)$$

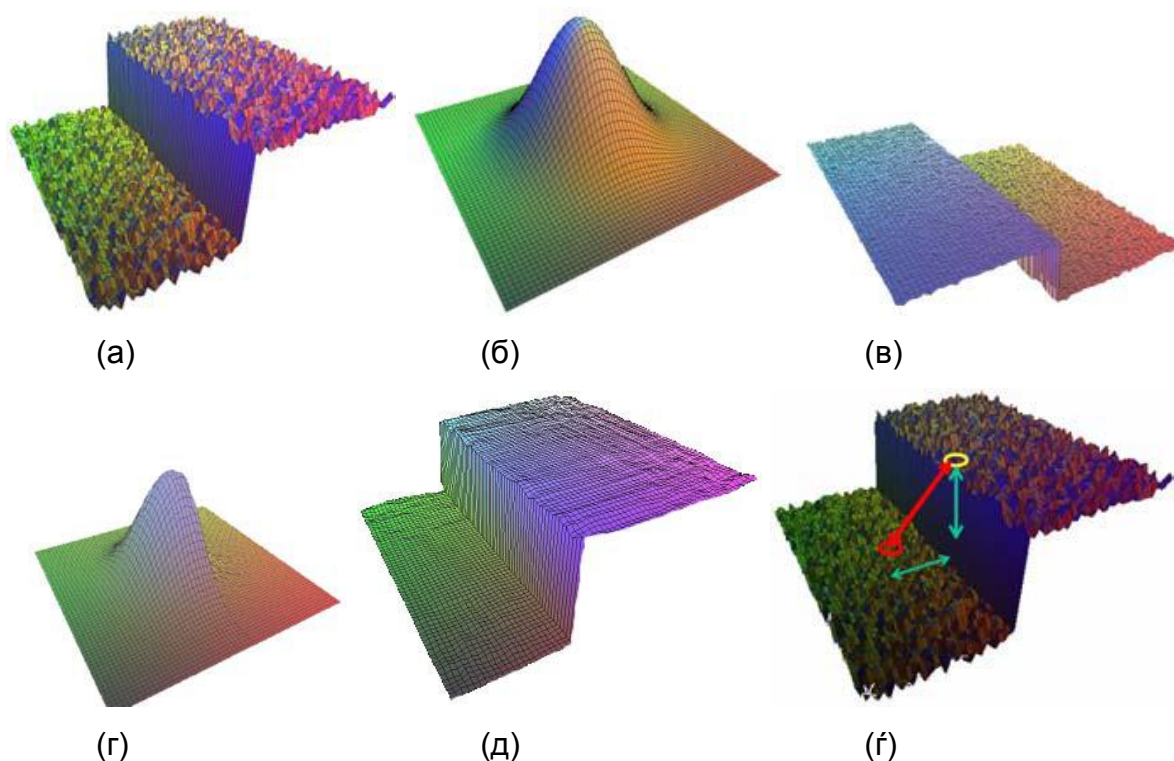
и опсег средина, слика 27г,

$$r(i, j, k, l) = \exp\left(-\frac{\|f(i, j) - f(k, l)\|^2}{2\sigma_r^2}\right), \quad (138)$$

Кога помножени заедно, тие ја даваат билатералната тежинска функција

$$\omega(i, j, k, l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2} - \frac{\|f(i, j) - f(k, l)\|^2}{2\sigma_r^2}\right). \quad (139)$$

Слика 28 покажува пример на билатерално филтрирање на шум на степенест раб.



Слика 28. Билатерално филтрирање : (а) влезен степенест раб со шум; (б) обласно филтрирање (гаусово); (в) опсежно филтрирање (сличност на вредност на централниот пиксел); (г) билатерално филтрирање; (д) излезен филтриран степенест раб; (е) 3-Д растојание помеѓу пиксели.

Иако билатералното филтрирање е доста споро во споредба со регуларното филтрирање на издвојување, бројни техники на забрзување се произведени за негово подобрување. Сепак овие техники тежнеат да користат повеќе меморија одколку регуларното филтрирање и поради тоа не се директно користат за филтрирање на слики во боја [3].

3.2.4 Пирамиди на слика

Често при трансформација на слика сакаме да ја промениме и резолуцијата на сликата. На пример, кај помала слика потребно е да ја промениме резолуцијата со цел таа да може да се претстави со сите детали на поголем екран. Или пак спротивно на ова, поголема слика, односно со поголема резолуција потребно е да ја намалиме со цел да зафаќа помалку простор во меморијата или да го убрза процесот на обработка на слика.

Понекогаш поради тоа што точно не се знае која е потребната големина на резолуција на слика, потребно е да се генерира цела *пирамида* од разни големини на слики, со цел при пребарување да се најде соодветната. Ќе разгледаме примери на гаусовата и лапласовата пирамида.

- Гаусова пирамида (намалување)

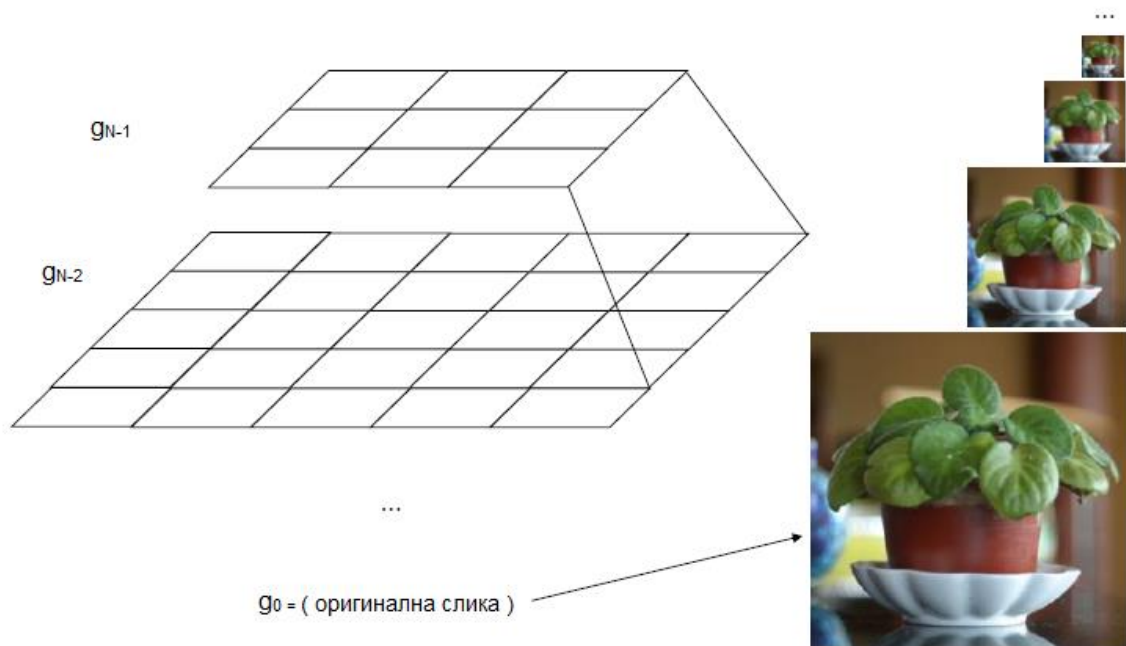
Целта е да се направи репрезентација на сликата со намалување и разложување на делови, од кои што треба да се извлечат карактеристични значајки, за да се намали шумот.



Слика 29. Оригиналната слика (g_0) и соодветно намалени репрезентации на оригиналната слика.

На слика 29 како влезен сегмент ја имаме оригиналната сликата со ознака (g_0) која што има големина $(2^N + 1) \times (2^N + 1)$, а како излезен сегмент ги добиваме соодветно намалени сликите (g_1, \dots, g_{N-1}). Од каде сликата (g_l) би имала големина $(2^{N-l} + 1) \times (2^{N-l} + 1)$.

Самата репрезентација на сликата може да биде претставена како пирамида од 3×3 , 5×5 , 9×9 , ..., $(2^N + 1) \times (2^N + 1)$ слики.



Репрезентацијата на сликата е базирана врз две основни операции:

1. *Измазнување* (анг. Smoothing) на сликата со филтри за мазнење, од кои што филтри секој филтер има двапати поголем радиус од претходниот.
2. *Намалување* (анг. Downsampling) на големината на сликата за половина по секое измазнување.

Овој процес е претставен со пример на слика 30 и равенките под неа за секој чекор последователно, како и на слика 31:



(a)



(б)



(в)



(г)

(д)

(е)

Слика 30. Прикажан е процесот на измазнување на оригиналната слика, без притоа да се намали големината на сликата.

$$\hat{g}_1 = w * g_0 \quad (140)$$

Каде што w е 5x5 филтер

$$\hat{g}_1(i, j) = \sum_{m=-2}^2 \sum_{n=-2}^2 w(m, n) * g_0(i - m, j - n) \quad (141)$$

$$\hat{g}_2 = w * \hat{g}_1 = w * (w * g_0) = (w * w) * g_0 \quad (142)$$

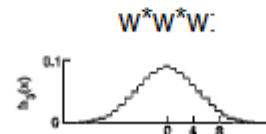
$$\hat{g}_3 = w * \hat{g}_2 = (w * w * w) * g_0 \quad (143)$$

$$\hat{g}_4 = w * \hat{g}_3 = (w * w * w * w) * g_0 \quad (144)$$

$$g_{l+1}(i, j) = \sum_{m=-2}^2 \sum_{n=-2}^2 w(m, n) * g_l(2i - m, 2j - n)$$



Каде што $(w * w)$ е всушност филтер чиј што радиус е двапати поголем од w , а $(w * w * w)$ пак е трипати.





Слика 31. Прикажани се чекорите на намалување за 1/2 на оригиналната слика после секое измазнување.

После секој чекор на измазнување последователно следува и намалување на сликата, бидејќи со самото измазнување се губат голем број на детали па е овозможено намалување на големината на сликата, а воедно и за да не се познава заматеноста која настанува.



(a)



(б)

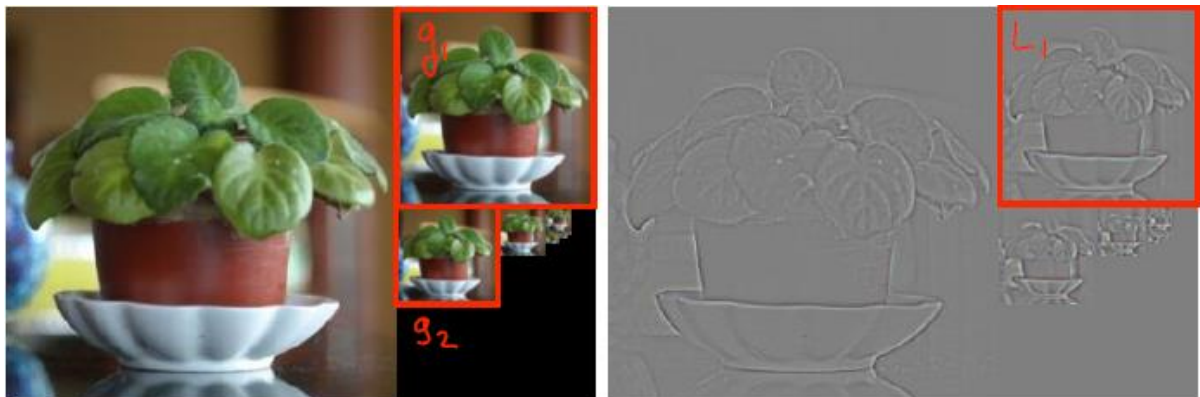


(в)

Слика 32. Детали кои што се изоставуваат при измазнувањето. (а) Оригинална слика, (б) слика измазнета, (в) детали кои не се пренесени во измазнувањето.

- Лапласова пирамида (зголемување)

Со помош на ова претставување може да го добиеме спротивниот процес од процесот на претставување на слика со гаусовата пирамида. Овде наместо измазнетите слики, се чуваат разликите добиени помеѓу оригиналната и измазнетата слика, односно деталите кои не се пренесени при измазнување. Овие разлики најчесто се во нијанси на сиво и зафаќаат помалку меморија.



Слика 33. Разликата L_1 при одземање на зголемена g_2 од g_1 .

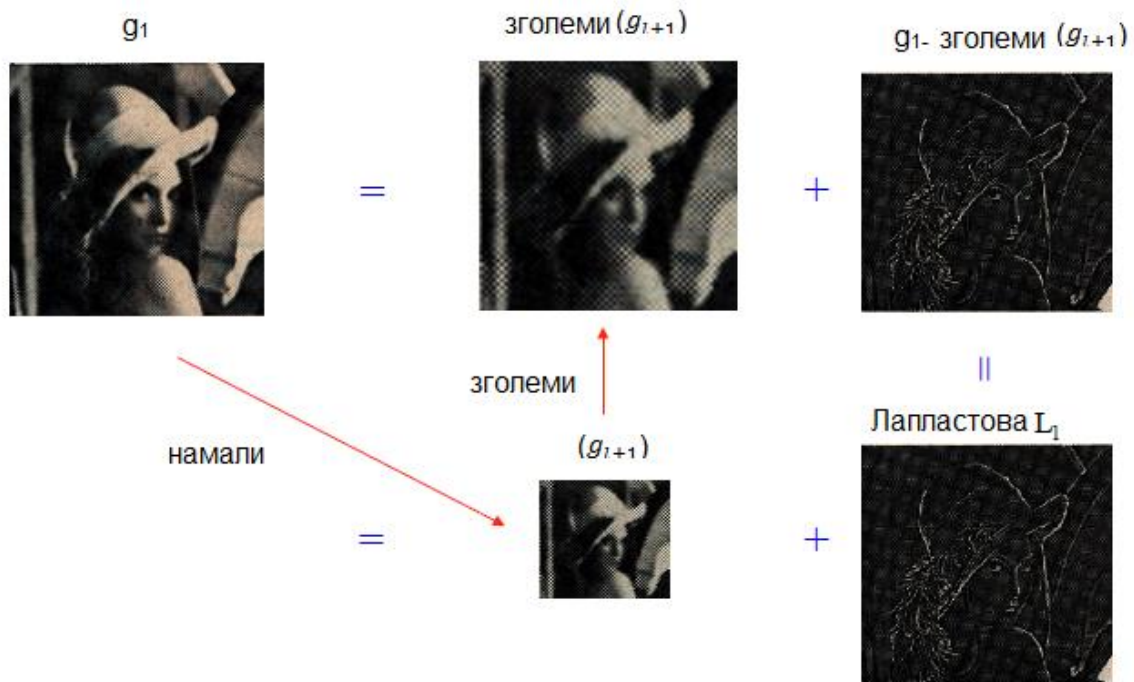
$$L_1 = g_1 - \text{зголеми}(g_2) \quad (145)$$

Со цел да го добиеме обратниот процес, односно зголемување од мала слика во поголема, потребно е да се направи споредба помеѓу малата слика g_{l+1} и

големата g_l . За да се направи споредбата мора да се изедначат двете големини на сликите, т.е. малата слика g_{l+1} треба да се зголеми до големина соодветна на g_l .

$$\text{зголеми}(g_l) = 4 \sum_{m=-2}^2 \sum_{n=-2}^2 w(m, n) * g_l\left(\frac{i-m}{2}, \frac{j-n}{2}\right) \quad (146)$$

Со оваа функција се зголемува нивото на сликата со дуплирање на големината од $(2^{N-l} + 1) \times (2^{N-l} + 1)$ во $(2^{N-l+1} + 1) \times (2^{N-l+1} + 1)$.



Слика 34. Процесот на Лапласова пирамида.

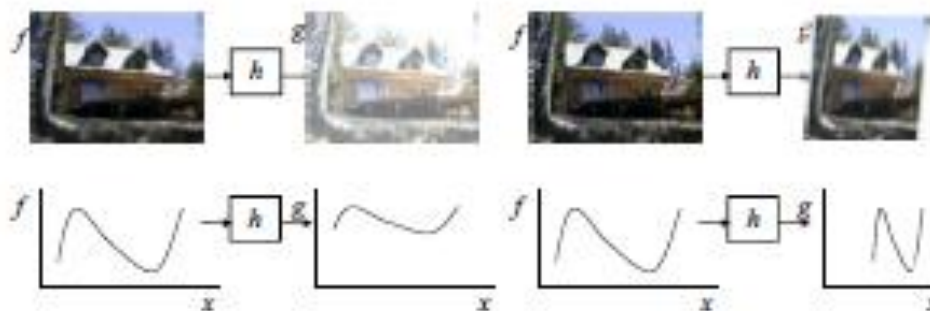
3.2.5 Геометриски трансформации

Знаеме дека со употреба на интерполација и деклинација може да се промени резолуцијата на слика. Сега ќе погледнеме како да изведеме по општа трансформација, како ротации или искривувања на слика [3]. За разлика од кај точкастите процеси што ги видовме погоре, каде што функцијата што се употребува на сликата го трансформира опсегот на сликата,

$$g(x) = h(f(x)), \quad (147)$$

Овде имаме функции кои што ја трансформираат областа,

$$g(x) = f(h(x)) \quad (148)$$






Слика 35. Искривување на слика содржи модифицирање на областа на функцијата на сликата, а не на опсегот.



Слика 36. Основни 2-Д трансформации на слика.

Параметарски трансформации на слика изведуваат глобални промени врз сликата, каде што однесувањето на трансформацијата е контролирано со мал број на параметри. На слика 36 има неколку примери на такви трансформации, кои се засновани на 2-Д геометриски трансформации. Формулите на овие трансформации се прикажани во табелата 1.

Трансформација	Матрица	#Длабочина	Зачувува	Ознака
Транслација	$[I t]_{2 \times 3}$	2	Ориентација	
Круто (еуклид)	$[R t]_{2 \times 3}$	3	Должина	
Слична	$[sR t]_{2 \times 3}$	4	Агли	
Сродна	$[A]_{2 \times 3}$	6	Паралелност	
Проекциска	$[\tilde{H}]_{3 \times 3}$	8	Прави линии	

Табела 1 Хиерархија на 2-Д трансформации на координати. Секоја трансформација ја зачувува не само особината која е наведена за неа туку и сите други кои се под неа.

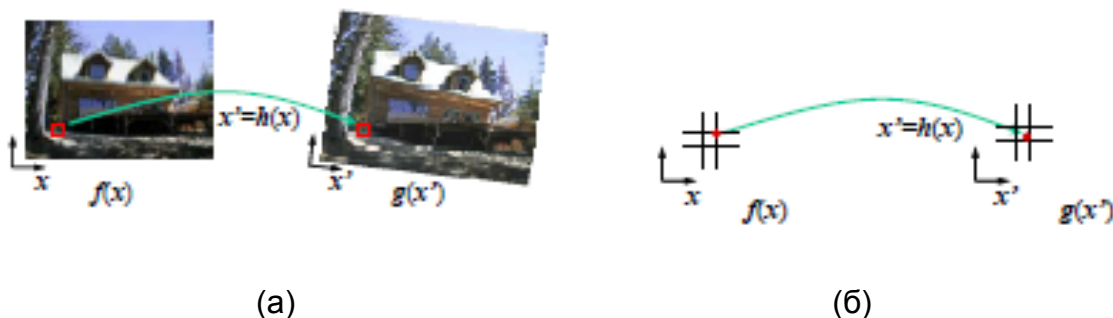
Во главно за дадена трансформација дефинирана со формула $x' = h(x)$ и слика $f(x)$, за да ги пресметаме вредностите на пикселите во новата слика $g(x)$, потребно е да се употреби алгоритам како следниов [3].

Procedure forwardWarp($f, h, \text{out } g$):

For every pixel x in $f(x)$

1. Compute the destination location $x' = h(x)$.
2. Copy the pixel $f(x)$ to $g(x')$

Алгоритам 1. Директен алгоритам за искривување на слика $f(x)$ и трансформирање во слика $g(x')$ преку параметарска трансформација $x' = h(x)$.



Слика 37. Директен алгоритам за искривување: (а) пиксел $f(x)$ е копиран на соодветната локација $x' = h(x)$ во слика $g(x')$; (б) детали од локацијата на пикселот.

Всушност овој пристап има неколку ограничувања. Процесот на копирање на пиксел $f(x)$ до локација x' во g не е добро дефиниран кога x' има вредност што не е број интегер. Во овој случај може да се заокружи вредноста на x' до најблиската координата со вредност интегер, па пикселот да се копира таму, но добиената слика има големи отстапувања и пиксели што не одговараат на оригиналните. Исто така може да се пренесе вредноста на пикселот до најблиските соседни пиксели во тежинска распределба, оставајќи притоа трага од тежината на секој пиксел и нормализација на крајот. Оваа техника се

нарекува попрскување (анг. *splatting*) и се користи понекогаш за волуменско рендерирање во графиката. За жал страда од големи отстапувања и доза на заматување. Вториот проблем со напредно искривување е појавата на дупки, поготово при зумирање на сликата. Пополнувањето на тие дупки со неговите соседни пиксели може да доведе до поголеми отстапувања и заматувања.

Што може да се направи е всушност е да се користи инверзно напредно искривување, каде секој пиксел во добиената слика $g(x')$ е превземен од оригиналната слика $f(x)$.

Ова се разликува од напредниот алгоритамот за искривување, по тоа што кога $\hat{h}(x')$ е предефиниран за сите пиксели во $g(x')$, повеќе немаме дупки. Уште побитно е тоа што проблемот со локација чија вредност не е интегер е надминат, и високо квалитетни филтри што го контролираат отстапувањето може да се користат [3].

Procedure inverseWarp($f, h, \text{out } g$):

For every pixel x' in $f(x)$

1. Compute the source location $x = \hat{h}(x')$.
2. Resample $f(x)$ at location x and copy to $g(x')$

Алгоритам 2. Алгоритам за инверзно искривување за креирање на слика $g(x')$ од слика $f(x)$ употребувајќи ги параметарските трансформации $x' = h(x)$.

Во други случаи, пожелно е да се формулира проблемот на искривување на слика како избирање на слика $f(x)$ со дадено мапирање $x = \hat{h}(x')$ од добиените пиксели x' до оригиналните пиксели x .

На пример, во оптичкиот ток го пресметуваме полето на проток (анг. flow field) како локација на изворниот пиксел кој го формира тековниот пиксел (пикселот на кој го пресметуваме токот), во спротивност со пресметката на дестинацискиот пиксел каде што оди пикселот. Слично кога вршиме корекција кај радијалната дисторзија, ја калибрираме леќата со пресметка за локацијата на секој пиксел во добиената слика, со локацијата на секој пиксел во оригиналната (дисторзираната) слика.



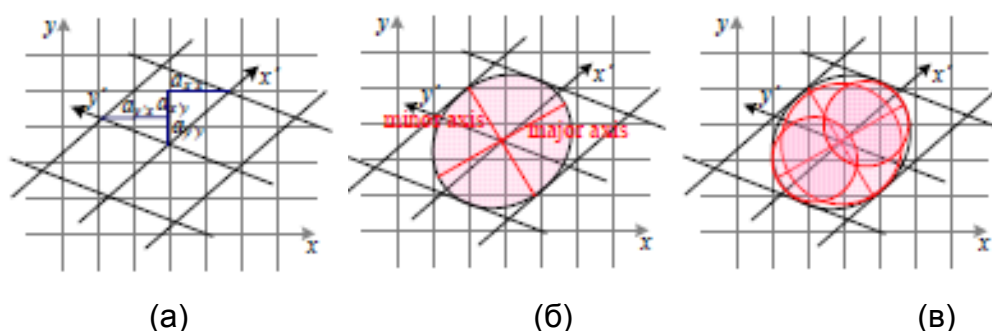
Слика 38. Инверзен алгоритам за искривување: (а) пиксел $g(x')$ е избран од соодветната локација $x = \hat{h}(x')$ во сликата $f(x)$; (б) детали од локациите на пикселот во оригиналната и добиената слика.

За пресметка на вредноста на $f(x)$ прикажана во неинтегер локација x , едноставно ќе го употребиме филтерот за избирање,

$$g(x, y) = \sum_{k,l} f(k, l) h(x - k, y - l), \quad (149)$$

Каде што (x, y) се под-пикселните вредности, а $h(x, y)$ е некоја основа за вметнување или измазнување.

Да претпоставиме трансформација со која што се врши ширење на x димензијата, а y димензијата се потиснува. Израмнувачката основа треба да изведува регуларни вметнувања по x димензијата, а измазнување по y димензијата. Ова станува уште покомплицирано во случај на поместена или перспективна трансформација.



Слика 39. Анизотропски филтрирање на текстура: (а) Јакобиева трансформација на A и индицираните хоризонтални и вертикални изборни релации $\{a_{x'x}, a_{x'y}, a_{y'x}, a_{y'y}\}$; (б) елипсоиден отпечаток од основа за измазнување; (в) анизотропско филтрирање со употреба на повеќе избори долж главната оска. Пикселите лежат на пресеците од линиите.

3.3 Одредување на значајки

3.3.1 Точки

Точкасти значајки може да се употребат за да се најде расфрлана множество (група) (ang. sparse) на локации кои што кореспондираат во различни слики, често како претходник за пресметување на поза, која што е всушност предуслов за пресметка на погусто множество на кореспонденции при употреба на стерео поклопување. Таква кореспонденција исто така може да се користи за порамнување на различни слики, т.е. кога се спојуваат слики како мозаици или кога се врши стабилизација на видео. Техниката на кореспондирачки (соодветни) значајки се употребува уште од почетокот на стерео поклопувањето, а сега се повеќе популарни кај апликациите за спојување на слики.

Постојат два главни пристапи за пронаоѓање на точкасти значајки и нивните соодветни коресподенти. Првиот пристап претставува техника на пронаоѓање на значајки во една слика кои што може точно да бидат следени употребувајќи локална техника за пребарување, како што е техниката на најмали квадрати. Вториот пристап е со независно детектирање на значајки во сите слики што се разгледуваат, а потоа споредба и соодветно составување на значајки базирано на нивниот изглед [3].

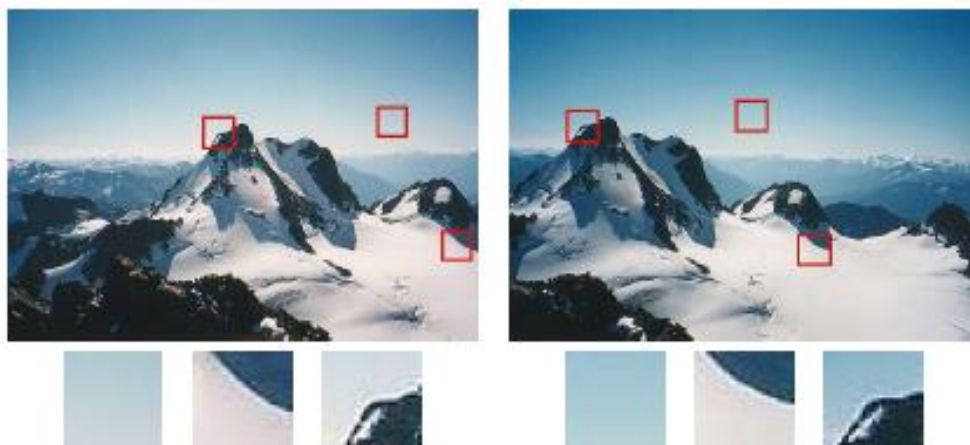
Првиот пристап е погоден кога сликите се снимани од блиски гледишта или со кратки прекини, додека вториот пристап е погоден кога се очекува големо движење или пак промена на изгледот. Значи имаме две спротивни стратегии. Една е со комплексна трансформација која ги опфаќа трансформациите на целата слика, додека другата е со едноставни трансформации на делови од сликата каде што може да се детектира движење. Најчесто се одбира моделот со локални трансформации, бидејќи глобалниот модел има бесконечно многу димензии.

Детекцијата на значајки ги делиме во четири посебни фази. Во фазата на *одредување на значајки*, секоја слика се пребарува за локации кои што ќе бидат лесно препознатливи и соодветно поврзани на други слики. Во фазата *идентификување на значајки*, секој регион околу одредени важни локации е претворен во компактен и стабилен идентификатор, што може да се поврзи соодветно со други идентификатори. Фазата на *поврзување на значајки*,

ефикасно пребарува слични кандидати кои што соодветно одговарат во други слики. Фазата на *следење на значајки* е алтернатива на претходната фаза, која што пребара само мали групи на соседни значајки околу тие што се одредени, па затоа е погоден за обработка на видео.

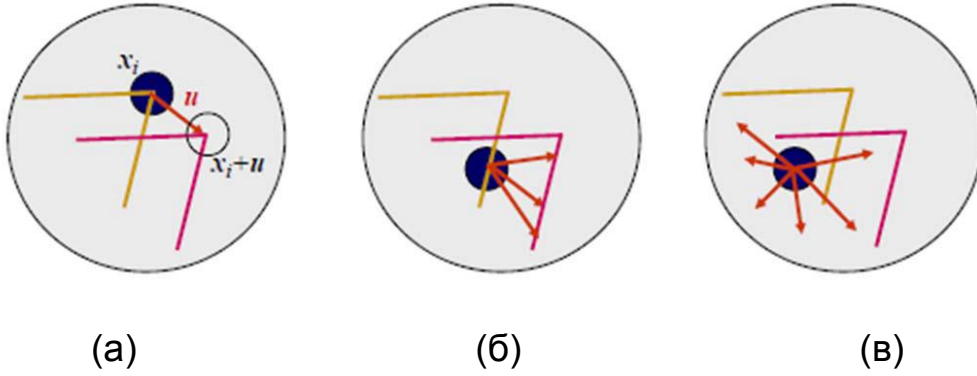
3.3.1.1 Детектори на значајки

За да го разгледаме проблемот на детекција на значајки да ја погледнеме слика 40. Може да се забележи дека деловите од сликата со поголем контраст можат полесно да се локализираат и следат, за разлика од деловите каде што нема контраст.



Слика 40. Сликата се поврзува со исечените делови под неа. Забележи како некои делови може да се лоцираат или поврзат со поголема прецизност од други

Деловите со поголем градиент можат полесно да се локализираат. Најлесно се локализира градиент во најмалку две различни насоки, како на слика 41.



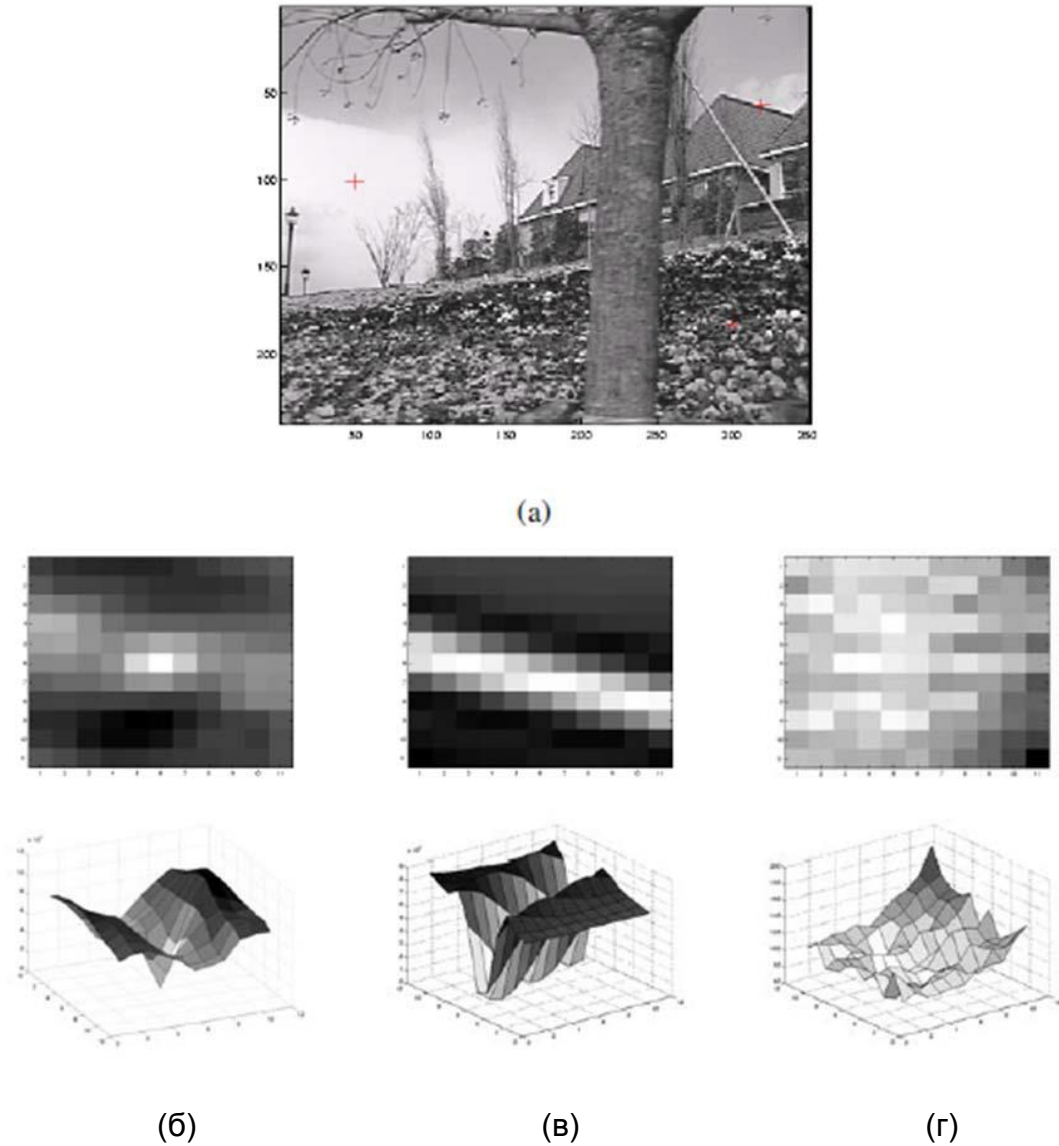
Слика 41. Проблеми на градиент за различни делови од слика: (а) стабилен проток; (б) класичен проблем на градиент; (в) регион без градиент.

$$E_{WSSD}(u) = \sum_i w(x_i)[I_1(x_i + u) - I_0(x_i)]^2 \quad (150)$$

Каде што I_1 и I_0 се двете слики кои се споредуваат, $u = (u, v)$ е векторот на поместување (анг. displacement), $w(x)$ е тежинска вредност за вредностите на сите пиксели во прозорецот кој што се споредува. Точноста на споредбата зависи и од големината на поместувањето Δu , која се користи во функцијата за автоматска корелација

$$E_{AC}(\Delta u) = \sum_i w(x_i)[I_0(x_i + \Delta u) - I_0(x_i)]^2 \quad (151)$$

Ако текстурата е униформна околу местото кое сакаме да го локализираме, тогаш локацијата која ја споредуваме се споредува самата со себе. Оваа појава се нарекува *авто корелација*.



Слика 42

Од слика 42. се гледаат разлчните ситуации кои може да настанат. Се гледа дека локацијата (в) има јак минимум па според тоа може добро да се локализира, додека локализацијата (г) нема стабилен минимум. Ако ја развиеме функцијата на сликата $I_0(x_i + \Delta u) \approx I_0(x_i) + \nabla I_0(x_i) \cdot \Delta u$ со користење на Тајлоровиот ред, ќе ја прецизно одредиме автокорелацијата како:

$$E_{AC}(\Delta u) = \sum_i w(x_i) [I_0(x_i + \Delta u) - I_0(x_i)]^2 \quad (152)$$

$$\approx \sum_i w(x_i) [I_0(x_i) + \nabla I_0(x_i) \cdot \Delta u - I_0(x_i)]^2 \quad (153)$$

$$= \sum_i w(x_i) [\nabla I_0(x_i) \cdot \Delta u]^2 \quad (154)$$

$$\Delta u^T A \Delta u, \quad (155)$$

Каде што :

$$\nabla I_0 = \left(\frac{\partial I_0}{\partial x}, \frac{\partial I_0}{\partial y} \right) (x_i) \quad (156)$$

претставува градинет на x_i . Матрицата за автокорелација може да се запише како:

$$A = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}, \quad (157)$$

Каде што w е тежинскиот фактор. Оваа матрица може да се интерпретира како тензор слика каде што надворешниот производ на градиенти ∇I_0 се конволуирани со тежинската функција w за да произведат естимација на иден пиксел од локалната (квадратна) функција за автокорелација.

- **Форстнер-Харис алгоритам**

Форстнер и Харис беа првите што предложија употреба на максима во ротациони ретки скаларни мерења изведени од матрицата на автокорелација за да се лоцираат главните точки за соодветно поклопување на сферни значајки. Оваа техника користи Гаусови тежински вредности, кои што прават детекторот да реагира интензивно на ротации на слика во рамнина.

Минималната eigenvalue вредност λ_0 , не е единствената вредност која што може да се употреби за да се најдат главните точки. Поедноставна вредност е

$$\det(A) - \alpha \text{trace}(A)^2 = \lambda_0 \lambda_1 - \alpha (\lambda_0 + \lambda_1)^2 \quad (158)$$

Со $\alpha = 0.06$, несвојствено за eigenvalue анализа, оваа вредност не бара употреба на квадратни корени, а сепак е ротационо непроменлива каде што $\lambda_1 \gg \lambda_0$. Исто така може да се користи и вредноста

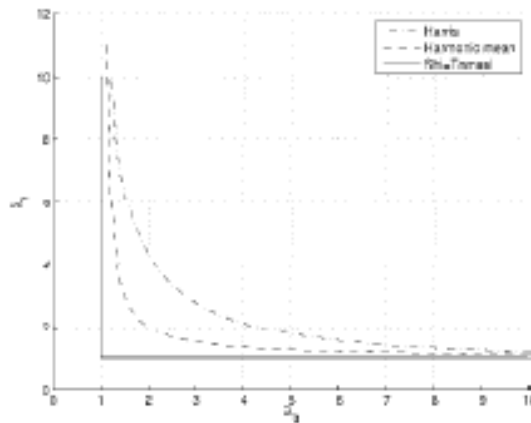
$$\lambda_0 - \alpha \lambda_1 \quad (159)$$

Која што исто така ја намалува реакцијата на 1Д кошиња, каде надоврзувачки грешки понекогаш влијаат врз помалите вредности.

Хармоничниот приказ е

$$\frac{\det A}{\text{tr } A} = \frac{\lambda_0 \lambda_1}{\lambda_0 + \lambda_1}, \quad (160)$$

И претставува подобра функција во регионот каде $\lambda_1 \approx \lambda_0$. Слика 4.7 покажува исоконтури од разни пресретнати оператори, од кои може да се забележи како две вредности се спојуваат за да се одреди крајната вредност.



Слика 43. Исоконтури од популарни функции за детекција на главни точки. Секој детектор бара точки каде двете вредностите λ_0, λ_1 од $A = w * \nabla I \nabla I^T$ се големи.

3.3.1.2 Опис на значајки

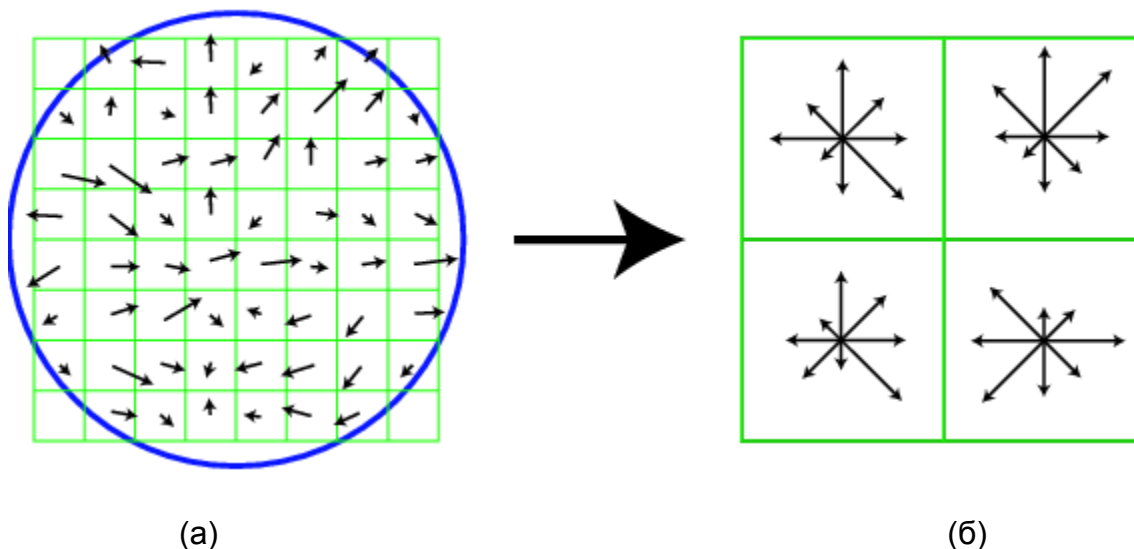
Од како ќе ги детектираме значајките, мора да ги поврземе сите значајки со соодветните локации во разни слики. Кај некои случаи на делови од видео или стерео делови кои се корегирани, локалното движење околу секоја точкаста значајка, најчесто е транслационо. Во овој случај може да се користат мерки со грешки, односно збирот на квадрирани разлики или нормализирани вкрстени корелации, за да се спореди интензитетот во мали делови околу секоја значајка. Поради тоа што одредувањето на локацијата на значајките може да биде непрецизно, а со тоа и локалното прикажувањето на нив ќе биде со промени во насоката и големината. И покрај исправувањето на овие промени, сепак локалното прикажување повторно ќе варира од слика до слика. Со цел за да се добијат подобри резултати, односно да се зачуваат промените при опишувањето на соодветните значајки, ќе го разгледаме следниов случај на опис на значајки.

- Непроменлива големина на значајка при трансформација – англ. Scale invariant feature transform (SIFT)

SIFT значајки се формираат со пресметување на градиентот на секој пиксел во 16x16 прозорец околу детектираните значајки, употребувајќи го правилното ниво од Гаусовата пирамида каде што се детектирани значајките. Големините на градиентот се намалуваат со Гаусовата функција за отфрлање, се со цел да се намали влијанието на градиентите што се далеку од центарот.

Во секој 4x4 квадрант се формира хистограм за насоката на градиентот, преку додавање на тежинската вредност на градиентот кон една од осумте насоки на хистограмот. За да се намалат ефектите на погрешна пресметка на локација и доминантна насока, секоја од 256 тежинските големини на градиентот е додадена кон 2x2x2 хистограм употребувајќи трилинеарна интерполација. Пренесувањето вредности кон хистограмот е всушност добра идеја во многу апликации каде што има пресметка на хистограми, т.е. за Хоуг трансформациите.

Добиените 128 позитивни вредности формираат грубата верзија на SIFT опишувачки вектор. За да се отстранат ефектите на контрасти (дополнителни варијации се веќе отстранети со градиентот), 128-Д векторот е нормализиран во единична должина. За дополнително да се намали опишувањето на други фотометрички варијации, вредностите се сечат до 0.2 и добиениот вектор е повторно нормализиран до единична вредност.



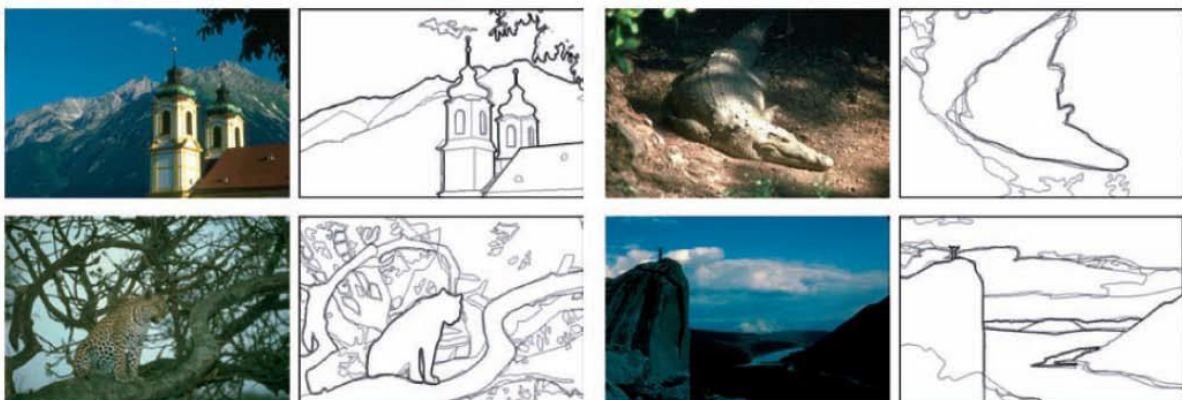
Слика 44. Шематски приказ на непроменлива големина на трансформирана значајка (SIFT): (а) Насоките и големините на градиентот се пресметани во секој пиксел и отстранети со Гаусовата функција. (б) Хистограм за тежинската насока на градиентот е направен во секој подреон употребувајќи трилинеарна интерполација.

3.3.2 Краеви и ќошиња

За разлика од точкастите значајки, кои беа доста корисни за наоѓање на локации во сликата и кои потоа можеа прецизно да се поврзат со соодветните локации во 2-Д, краевите и ќошињата како значајки се многу посликовити и пренесуваат доста интересни шематски асоцијации. На пример границите на објектите, што исто така одговараат со 3-Д настаните, се означени со видливи контури. Друг вид на краеве одговараа на сенките на границите, каде површината се менува често. Изолирани точкасти значајки често може да се групират во поголеми криви или конури, како и во прави линии. Ова го олеснува препознавањето на објектите и контурите, иако се нацртани само со едноставни линии.

3.3.2.1 Детекција на краеве и ќошиња

Да ја погледнеме сликата 45 и да ги забележимо најистакнатите краеве (рабови) или граници на објекти.



Слика 45. Човекова детекција на граници. Затемнетите краеве одговараат на најчестите локации каде луѓето мислат дека има објекти.

Најчесто краеве и објекти се детектираат помеѓу реони со различна боја, интензитет или исполнетост на објектите. Сепак да се подели слика на објекти во неконтрастна средина е тешка задача. Поради такви услови најдобар пристап е да се дефинираат краеве како локацијата каде има нагла промена на интензитетот.

Да ја замислиме сликата како висинско поле (анг. height field). На таква површина краевите се појавуваат на места од стрмни падини или на реони од блиски контурни линии (како на топографска мапа). Математички начин да се дефинира падината и насоката на површината е преку нејзиниот градиент

$$J(x) = \nabla I(x) = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right) (x). \quad (161)$$

Локалниот градиент на векторот J покажува кон насоката на најголемата стрмнина во функцијата на интензитет. Големината е индикатор на падината или јачината на варијациите, додека ориентацијата покажува кон насока нормална на локалната контура.

Одредувањето на деривациите на слика подразбира високи фреквенции, поради кои се зголемува и шумот, па поради пропорцијата на шум со сигнал, при големи фреквенции на сигнал имаме и големи шумови. Заради тоа сликата мора да се измазни со ниско пропусен филтер, пред да се пресмета градиентот. Бидејќи сакаме да користиме детекторот на краеве и кошиња кој ќе биде независен од насоката, пожелно е да користиме кружно симетричен филтер за измазнување. Гаусовиот филтер е независен кружно симетричен филтер, па се употребува во многу алгоритми за детекција на краеве и кошиња.

Диференцијацијата е линеарна операција, па затоа се меша со другите линеарни операции за филтрирање. Градиентот на измазнетата слика може да се запише како

$$J_{\sigma}(x) = \nabla[G_{\sigma}(x) * I(x)] = [\nabla G_{\sigma}](x) * I(x), \quad (162)$$

Па можеме да ги додадеме на сликата, хоризонталните и вертикалните изводи од Гаусовата основна функција,

$$\nabla G_{\sigma}(x) = \left(\frac{\partial G_{\sigma}}{\partial x}, \frac{\partial G_{\sigma}}{\partial y} \right) (x) = [-x, -y] \frac{1}{\sigma^3} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \quad (163)$$

Параметарот σ ја означува ширината на Гаусовата функција.

За многу апликации потребно е намалување на овој градиент на слика, се со цел да се добие повратна информација за само изолирани краеве, односно како единствени пиксели на уникатни локации долж краевите на една контура. Ова може да се постигне со барање на *максима* во големината на градиентот во насока нормална на насоката на работ. Наоѓањето на овие максимални вредности соодветствува со земање и насочување на насоката на јачината на површината кон насоката на градиентот и при тоа барајќи да нема пресеци. Посакуваната промена на насоката е еквивалентна со скаларен производ помеѓу вториот оператор на градиентот и резултатот од првиот

$$S_{\sigma}(x) = \nabla J_{\sigma}(x) = [\nabla^2 G_{\sigma}(x) * I(x)] \quad (164)$$

Скаларниот производ на операторите на градиентот со градиентот се нарекува Лапласов. Добиениот кернел

$$\nabla^2 G_{\sigma}(x) = \frac{1}{\sigma^3} \left(2 - \frac{x^2 + y^2}{2\sigma^2} \right) \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \quad (165)$$

Затоа се нарекува Лапласов од Гаусов (анг. Laplacian of Gaussian LoG) кернел. Овој кернел може да се подели на два посебни дела,

$$\nabla^2 G_{\sigma}(x) = \frac{1}{\sigma^3} \left(1 - \frac{x^2}{2\sigma^2} \right) G_{\sigma}(x) G_{\sigma}(y) + \frac{1}{\sigma^3} \left(1 - \frac{y^2}{2\sigma^2} \right) G_{\sigma}(y) G_{\sigma}(x) \quad (166)$$

затоа што се дозволува поголема ефикасна примена употребувајќи разделени филтри.

Во пракса често LoG мешањето, се заменува со разлики на Гаусово мешање (анг. Difference of Gaussian DoG), бидејќи формите на кернелот се поквалитетни. Ова е особено применливо доколку Лапласовата пирамида е пресметана.

Всушност не е секогаш потребно да се земаат разлики помеѓу подесувачките нивоа кога се пресметуваат краевите на површината. Кога кернелот е помал тогаш шумот е намален. Додека поголем кернел претставува пресметка на интензитетот врз поголем реон. Поради тоа, кога и да DoG слика го промени знакот, доаѓа до заматување на сликата, односно од потемна слика преминува во значително посветла слика.

Од како го пресметавме знакот на функцијата $S(x)$, потребно е да ги најдеме местата каде што нема пресеци и да ги претвориме во елементи на краеве (анг. *edges*). Лесен начин да се детектираат и претстават местата каде што нема пресеци е да се бараат на подесените локации на пиксели x_i и y_i каде што знакот менува вредност, $[S(x_i) > 0] \neq [S(x_j) > 0]$.

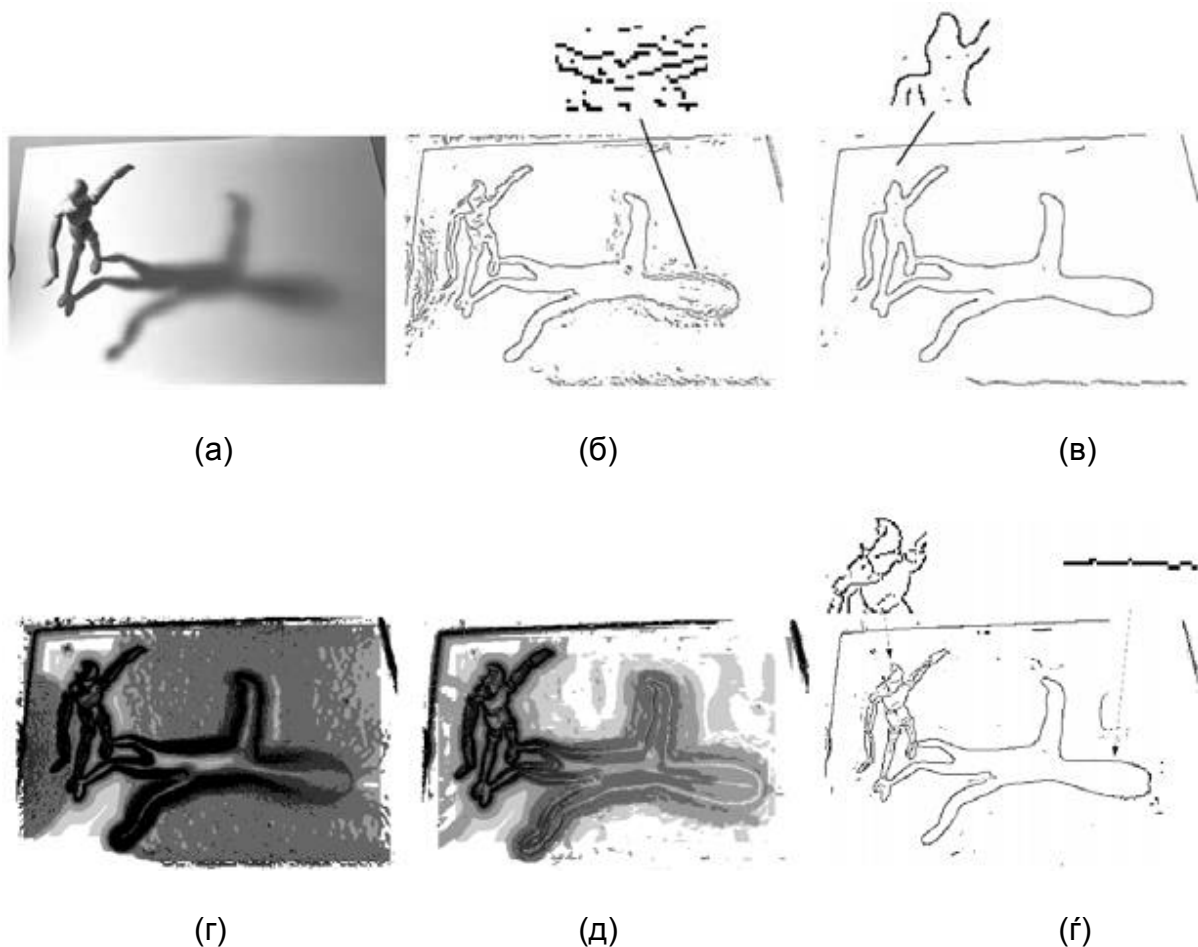
Субпозицијата на пикселот од овој пресек може да се добие со пресметување на x – пресекот од линијата што ја спојува $S(x_i)$ и $S(x_j)$,

$$x_z = \frac{x_i S(x_j) - x_j S(x_i)}{S(x_j) - S(x_i)}. \quad (167)$$

Ориентацијата и јачината на такви краеве може да се добие со линеарна интерполација на вредностите на градиентот пресметани на оригиналната пикселна мрежа.

Алтернативна репрезентација на краевите може да се добие со поврзување на подесувачки краеве на матрицата за да се добијат краеве што се наоѓаат внатре во секој квадрат, што е формиран со четири подесувачки пиксели во оригиналната пикселна мрежа. Придобивката од ваквиот приказ е тоа што краевите сега се наоѓаат за пола пиксел поместени од оригиналната пикселна мрежа и се полесни за пристапување и зачувување. Како и пред ова, ориентацијата и јачината на краевите може да се добие со интерполација на градиентот или преку пресметка на овие вредности од DoG сликата.

Во апликации каде што прецизноста на ориентацијата на краевите е доста важна, може да се употребат повеќе-нивоа управувачки филтри. Со таквите филтри се прави поголема селекција на продолжени краеве и исто така се има можноста за подобро претставување на кривите пресеци, бидејќи тие можат да претстават повеќе ориентации на еден ист пиксел. Нивната негативност е тоа што тие се поскапи (позахтевни) а воедно и решението за добиената насока за јачината на краевите нема едноставна форма.



Слика 46 Селекција за детектирање на краеве: (а) оригинална слика; (б-в) Кани детекција на краеве, доведена до мали размери; (г) минимална скала за проценка на градиентот; (д) минимална скала за втора изведена пресметка; (е) последно детектирани краеве.

3.3.3 Линии

Додека краевите и општите криви линии се погодни за прикажување на контурите на природните објекти, свет создаден од човекот е полн со прави линии. Детектирањето и соодветното поврзување на овие линии, може да биде доста корисно во многу апликации, како што се моделирање на архитектура, проценка на поза во урбано населени места и анализа на испечатен документ.

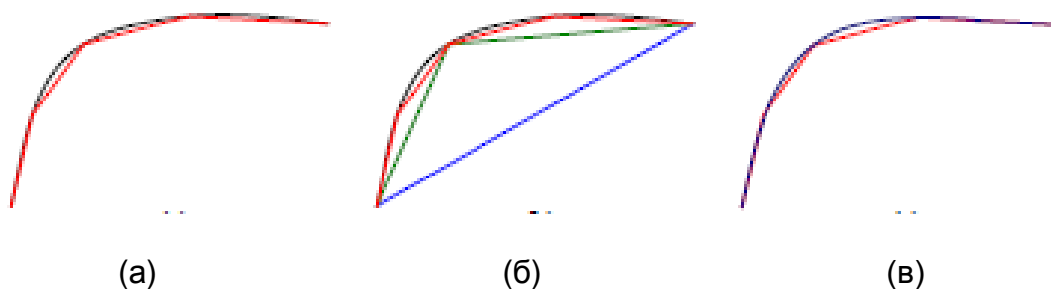
Овде ќе разгледаме некои техники за разложување линеарни описи на специфични делови од криви линии. Ќе го разгледаме алгоритмот за апроксимација на крива и Хугх трансформацијата, којашто може да се користи

за групирање на краеве во линиски сегменти дури и да има прекини или осцилации.

3.3.3.1 Последователни апроксимации

Знаеме дека опишувањето на криви линии како серија од 2-Д локации $x_i = x(s_i)$ обезбедува општ приказ доволен за соодветно совпаѓање и понатамошно процесирање. Во многу апликации е пожелно да се апроксимира таква крива со поедноставен приказ, т.е. како делови од (анг. Piecewise) линеарна линија.

Повеќе техники се имаат развиено за да се изведи оваа апроксимација, која е уште позната како поедноставување на линија. Една од најстарите и наједноставните е прикажана на слика 47, која што последователно ја дели линијата во точка која е најдалеку од правата што ги спојува двата краја на линијата.



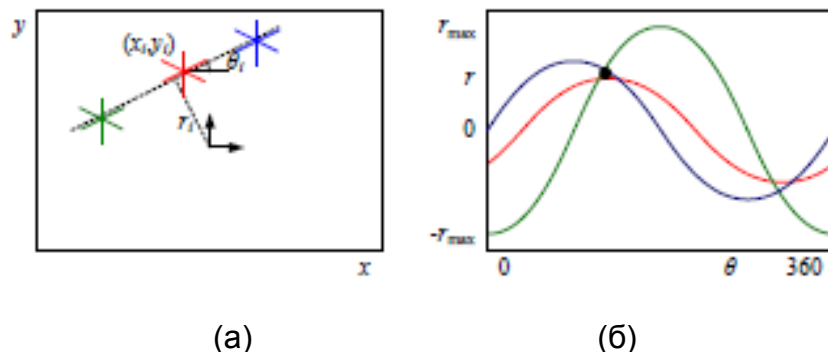
Слика 47. Апроксимација на крива како поврзани отсечки: (а) оригинална крива и поврзани отсечки со црвена боја; (б) последователна апроксимација со наоѓање на најодалечената точка од моменталната апроксимација; (в) блага интерполација

Откако поедноставувањето на линијата е направено, може да се употреби за апроксимација на оригиналната крива линија.

3.3.3.2 Хугх трансформација

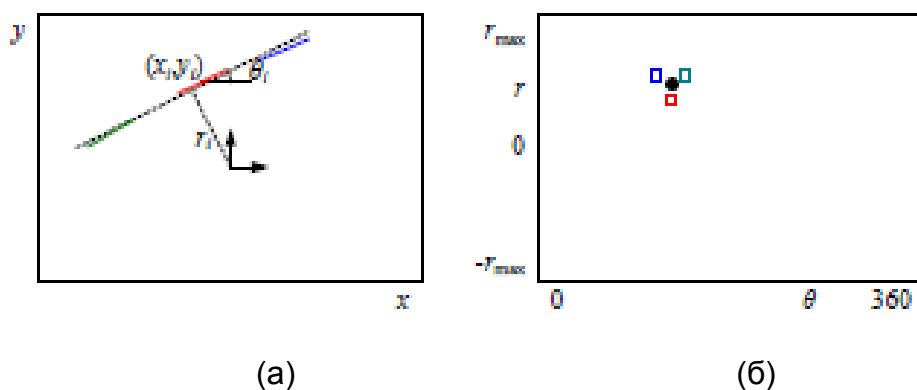
Апроксимација на крива со линија од отсечки може да доведе до успешно добивање на линија, но линиите во светот понекогаш се непрекинати или

направени од многу колинеарни линиски сегменти. Во повеќето случаеви пожелно е да се групираат тие колинеарни линиски сегменти во продолжена линија.



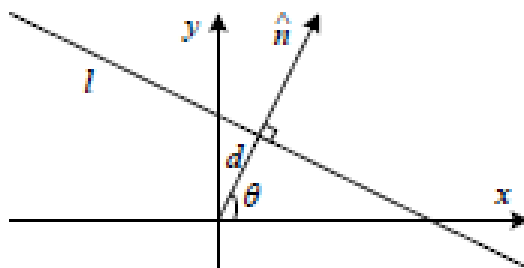
Слика 48. Оригинална хугх трансформација: (а) секоја точка гласа за комплетна потенцијална линија $r_i(\theta) = x_i \cos \theta + y_i \sin \theta$; (б) секоја линија прави синусоида во (r, θ) ; нивниот пресек прави посакувана равенка на линија.

Хугх трансформацијата е доста добра техника со која краевите и кошињата допринесуваат за создавање на веродостојни локации на линија. Во оригиналната формулација на слика 48, секое коше покажува за сите можни линии што минуваат низ него, а линии кои соодветствуваат на надоврзување со него со цел да се создаде линија се разгледуваат. Доколку точките на линијата се навистина точки, подобар пристап е да се добие информација за локална ориентација на секој крај за да се добие една акумулирана ќелија, како на слика 49.



Слика 49. Ориентациона Хугх трансформација: (а) крај параметаризиран со (r, θ) координати, со $\hat{n}_i = (\cos \theta_i, \sin \theta_i)$ и $r_i = \hat{n}_i \cdot x_i$; (б) (r, θ)

надоврзувачка низа, покажувајќи ги гласовите за трите краеви обележани со црвена, зелена и сина.



Слика 50. 2-Д равенка на линија изразена во услов на нормална \hat{n} и растојанието до центарот d .

Хибридна стратегија каде секој крај гласа за број на можни насоки или локации што се наоѓаат близу пресметаната насока.

Пред да гласаме за линиски хипотези, мора да избереме пригоден приказ. Слика 50 го покажува нормалното растојание (\hat{n}, d) параметризирано за линија. Бидејќи линиите се составени од сегменти, важи дека нормална линија \hat{n} покажува кон иста насока како и градиентот на сликата $J(x) = \nabla I(x)$. За да добиеме минимални два параметри за приказ на линии, го конвертираме нормалниот вектор во агол

$$\theta = \tan^{-1} n_y/n_x, \quad (168)$$

Како што е прикажано на слика 50. Опсегот на можни (θ, d) вредности е $[-180^\circ, 180^\circ] \times [-\sqrt{2}, \sqrt{2}]$, претпоставувајќи дека употребуваме нормализирани координати на пиксели што лежат на $[-1, 1]$. Бројот на цртки што ќе се употребуваат на секоја од оските зависи од прецизноста на позицијата и насоката на секој крај и густината на линијата, а тоа најдобро се добива при неколку тестирања.

При дадена параметризација, Хугх трансформацијата продолжува како што е прикажано во алгоритмот подолу.

Procedure Hough ($\{x, y, \theta\}$)

1. Clear the accumulator array.
2. For each edgel at location (x, y) and orientation $\theta = \tan^{-1} n_y/n_x$, compute the value of $d = xn_x + yn_y$ and increment the accumulator corresponding to (θ, d) .
3. Find the peaks in the accumulator corresponding to lines.
4. Optionally re-fit the lines to the constituent edgels.

Забележуваме дека оригиналната формулација на Хугх трансформацијата, која што не бара насоката θ на крајот да е позната, има дополнителен циклус во вториот чекор што поминува преку сите можни вредности на θ и врши зголемување на цела серија на акумулатори.

Постојат многу детали за Хугх трансформацијата да работи добро, но овие се најдобри и добиени со тестирања и имплементации.

Алтернатива на 2-Д приказот (θ, d) на линии е да се употреби целосната 3-Д $m = (\hat{n}, d)$ равенка на линија, проектирана на сфера. Додека сферата може да се параметаризира користејќи сферни координати,

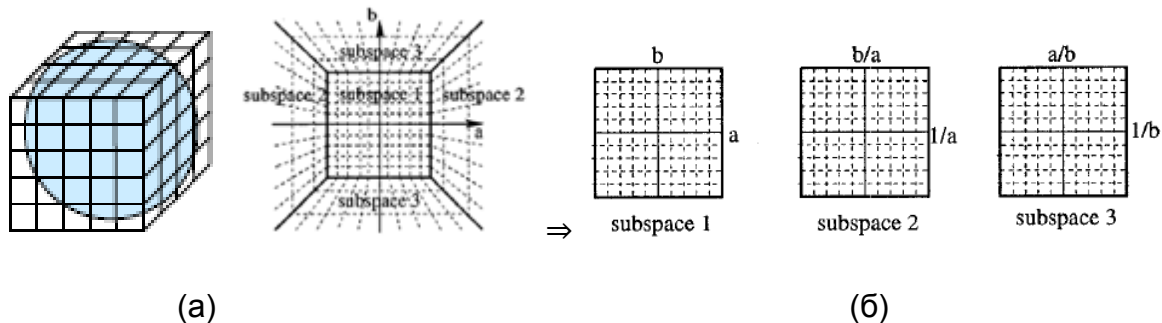
$$\hat{m} = (\cos\theta\cos\varphi, \sin\theta\cos\varphi, \sin\varphi), \quad (169)$$

Сепак ова не ја поедноставува сферата и сеуште мора да се користи тригонометрија [3].

Алтернативна репрезентација може да се добие со употреба на коцка мапа, т.е. проектирање на m врз коцка, слика 51а. за да се пресметаат координатите во коцка мапа на 3-Д вектор m , прво се наоѓа најголемата компонента на m т.е. $m = \pm \max(|n_x|, |n_y|, |d|)$, и употребувајќи го ова за да се селектира една од шесте страни на коцката. Поделувањето на преостанатите координати со m и дозволува користење како индекси за страните на коцката. Со ова се избегнува тригонометријата, но бара одлучувачка логика.

Една предност на употребата на коцка мапа, е тоа што сите линии минуваат низ точка која кореспондира со линиските сегменти од страните на коцката. Ова е корисно ако оригиналната верзија на Хугх трансформацијата се користи. Равенката на линија се прикажува како $ax + b + y = 0$, што всушност x и y оските

ги третира како несиметрични. Забележуваме дека ако го ограничимо $d \geq 0$ со игнорирање на знакот на градиентот, може да користиме половина коцка, што може да се претстави употребувајќи само три страни, како на слика 51б.



Слика 51 Репрезентација на коцка мапа на равенки на линија: (а) коцка мапа околу сферата; (б) проекција на половина коцка на трите подпростори.

4. Computer Vision Toolbox во Матлаб

4.1 Главни карактеристики

Computer Vision Toolbox во Матлаб претставува пакет на алатки за компјутерска визија кој нуди алгоритми, функции и апликации за дизајнирање и симулирање на компјутерска визија и системи за обработка на видеа. Може да се изведува детекција на значајки, извлекување, соодветно поврзување, може да се детектира објект и да се следи неговото движење, како и обработка на видео. За 3-Д компјутерската визија, пакетот поддржува калибрација на камера, стерео визија, како и реконструкција на 3-Д сцени. Со помош на алгоритми кои што се базирани на изучување, може да се подобри и извежба детекцијата на објекти, препознавањето на објект и системи за наоѓање на слики. Алгоритмите се достапни како Матлаб функции, системски објекти и блокови во Симулинк [6]. За дизајнирање вградени системи, пакетот со алатки подржува аритметика на фиксирана точка и генерирање на ц-код.

Главни карактеристики

- Детекција и следење на објект, со вклучени Viola-Jones, Kanade-Lucas-Tomasi (KLT), и Kalman методите на филтрирање

- Подобрување на детекција на објект, препознавање на објекти и системи за наоѓање на слики
- Калибрација на единечна и стерео камера, вклучувајќи и детекција на каскадни објекти и апликација за автоматска работа
- Стерео визија, со вклучена корекција, пресметка на разлика и 3-Д реконструкција
- Обработка на 3-Д точки, со I/O, визуализација, регистрација, отстранување на шум и геометриско вклопување
- Детекција на значајки, извлекување и соодветно совпаѓање
- Поддршка за генерирање на ц-код и аритметика на фиксирана точка со производи на генерализација

4.2 Одредување на карактеристични точки (значајки)

Локалните значајки и нивните идентификатори се главните блокови на многу алгоритми на компјутерска визија. Нивните апликации вклучуваат регистрирање на слика, детекција и класифицирање на објект, следење и пресметување на движење. Овие алгоритми користат локални значајки за подобро управување со скалирање, ротирање и оклузија (анг. occlusion). Алгоритмите во Computer Vision Toolbox пакетот содржат FAST, Harris, и Shi & Tomasi детектори на краеве и ќошиња, и SURF и MSER детектори на дупки. Пакетот со алатки ги вклучува SURF, FREAK, BRISK, LBP, и HOG идентификатори. Може да се мешаат и поврзат детекторите и идентификаторите во зависност од потребите на апликацијата.



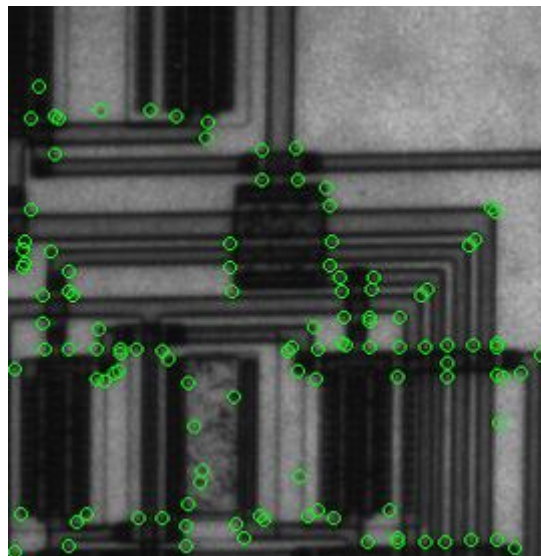
Слика 52. SURF (лево), MSER (средина) и детекција на краеве (десно) со системот пакет на алатки за компјутерска визија. Употребувајќи иста слика, три различни видови на значајки се детектирани и резултатите се поставени врз оригиналната слика.

4.2.1 Локални значајки

Локалните значајки се всушност патека или одвоена структура најдена во внатрешноста на слика, како точка, ќоше, раб или мал дел од слика. Тие се обично поврзани со дел од слика што се разликува од соседната околина преку текстурата, бојата или интензитетот. Што всушност значајката претставува не е толку значајно, само тоа дека се разликува од околината каде што се наоѓа. Примери на локални значајки се пиксели на дупки, ќошиња и краеви [6].

Пример за детекција на ќоше или раб во Матлаб:

```
I = imread('circuit.tif');  
corners = detectFASTFeatures(I,'MinContrast',0.1);  
J = insertMarker(I,corners,'circle');  
imshow(J);
```



Слика 53.

Детектори што се базирани на градиентот и пристапот кон варирањето на интензитетот, детектираат добри локални значајки. Овие значајки вклучуваат краеви, ќошиња, дупки и реони. Добра локална значајка ги има следниве параметри:

Повторувачки детекции:

- Кога се дадени две слики од иста сцена, повеќето значајки кои се најдени од страна на детекторот во двете слики се исти. Значајките се инертни на промени на поглед и шумови.
- Карактеристични
- Соседството околу центарот на значајката, варира доволно за да дозволи веродостојна споредба помеѓу значајките.
- Локализирачки
- Значајката е врзана со уникатна локација. Промени во погледот не ја променуваат локацијата.

4.2.2 Употреба на значајки

Регистрирањето на две слики е едноставен начин за да се разберат локалните значајки. Овој пример наоѓа геометриска трансформација помеѓу две слики. Употребува локални значајки за да пронајде добро лоцирани маркантни точки.

Пример прикажани се две слики.

Првата слика е оригиналната слика, слика 54.

```
original = imread('cameraman.tif');
figure;
imshow(original);
```



Слика 54. Оригинална слика

Втората слика е оригиналната слика само ротирана и скалирана, слика 55.

```
scale = 1.3;
J = imresize(original,scale);
```

```
theta = 31;
distorted = imrotate(J,theta);
figure
imshow(distorted)
```



Слика 55. Ротирана и скалирана слика1

4.3 Калибрирање на камера

Може да се користи калибрација на камера за да се пресметат внатрешните и надворешните параметри на камерата, како и параметрите за дисторзија на леќата. Потоа параметрите на камерата може да се користат во многу апликации на компјутерска визија. Овие апликации вклучуваат отстранување на ефекти на дисторзија на леќа од слика, мерење на објекти или реконструкција на 3-Д сцени снимани од повеќе камери [6].



За калибрирање на камерата во Computer Vision Toolbox преку апликацијата следи:

- Подготви слики, камера и шаблон за калибрирање
- Внеси слики
- Калибрација на камерата
- Проценка на точноста на калибрацијата
- Подеси параметри за подобрување на точноста
- Изнеси ги параметрите на објектот

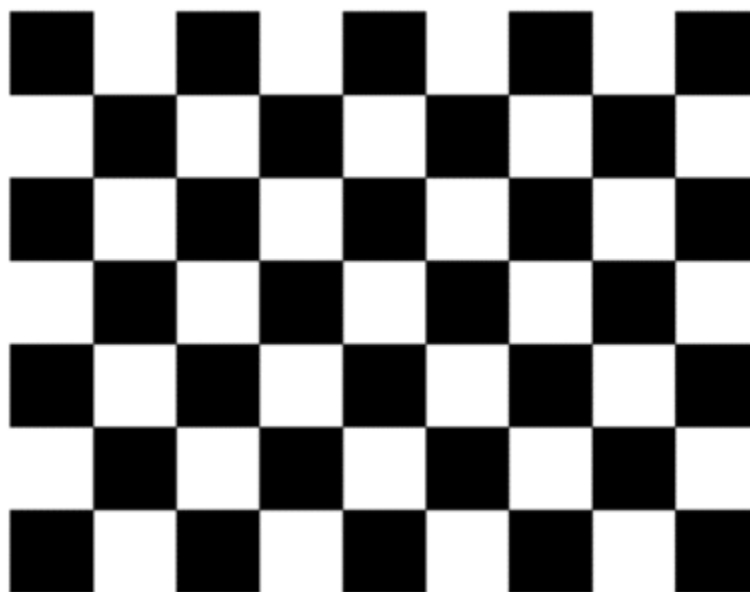
Во некои случаи, првите резултати работат одлично, па нема потреба од дополнителни подесувања пред да се изнесат параметрите. Ако е потребно да

се направат подесувања, може да се искористи функцијата во матлаб за калибрирање на камера.

Калибрацијата на камерата е важен чекор во компјутерската визија особено ако сакаме да извлечеме метрички информации од 3Д просторот преку 2Д слики. Има неколку техники кои се занимаваат со оваа проблематика и цело време се развиваат нови. Најпознати се: калибрација базирана на 3Д објект, калибрација базирана на 2Д рамнина, 1Д калибрација базирана на линија и самокалибрација.

Со калибрацијата се пресметуваат параметрите на леќата и на светлинскиот сензор на камерата. Со овие параметри потоа може да се корегираат дисторзијата на леќата, можат да се мерат објекти во реални мерни единици директно од слика или точно да се одреди позицијата на камерата во просторот. Овие задачи се користат во апликации слични на нашата, а освен тоа се користат во апликации за машинско учење, во роботиката, за навигациски системи, 3Д реконструкција и сл.

Параметрите на камерата вклучуваат внатрешни (интринсични), надворешни (екстринсични) и коефициенти на дисторзија. За да се добијат овие параметри ни требаат 3Д точки во просторот и нивни 2Д соодветни слики. За калибрација обично се користат шаблони со познати димезии, најпознат меѓу нив е шаблонот со шаховско поле, слика 56.

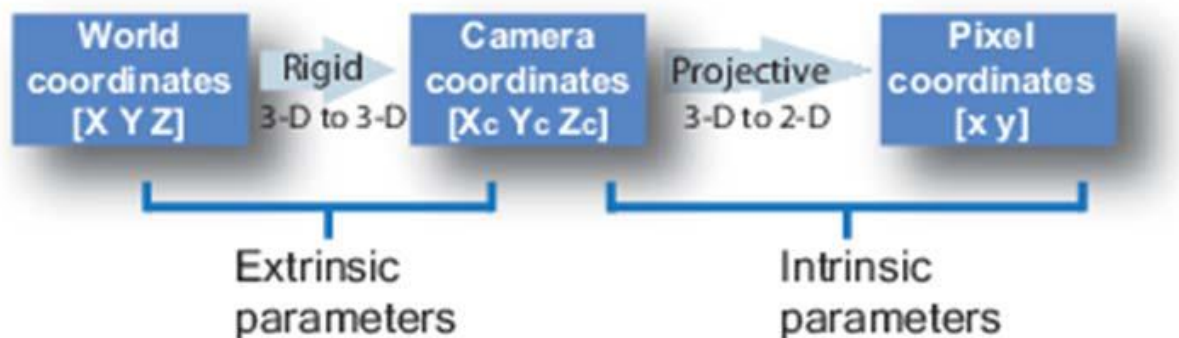


Слика 56. Шаблон на шаховско поле

Треба да ги знаеме димензиите на полињата како и димензиите на целиот шаблон, а формата на шаблонот треба да е правоаголник, т.е. да има парен број на редови, а непарен број на колони или обратно. Овие информации ни се потребни за калибрацискиот софтвер. Шаблонот се печати и треба да се сликаат одреден број на слики, од 10 до 20 слики, преку кои со калибрациски софтвер ги добиваме параметрите. Калибрациските алгоритми обично користат два модели за калибрација на камера, и тоа:

- Моделот на камера без леќа, и
- Модел на камера со дисторзија на леќата.

Ние го користиме првиот модел. Ове е недостаток на алгоритмот бидејќи дисторзијата на леќата влијае во голема мера во 3-Д реконструкцијата на просторот, особено кога се користи во надворешни услови кога имаме поблиски и подалечни објекти. Алгоритмот за калибрација ја пресметува матрицата на камерата со користење на надворешните и внатрешните параметри. Надворешните параметри ја претставуваат трансформацијата од 3-Д глобален координатен систем во 3-Д координатен систем на камерата. Внатрешните параметри ја претставуваат трансформацијата од 3-Д координати на камерата во 2-Д координати на слика, слика 57.



Слика 57. Трансформација од глобални во координати на пиксели.

Внатрешните параметри се всушност тие што ни требаат, тие ги сочинуваат фокалната должина, координати на оптичкиот центар, и коефициентите на закосеност, слика 58. Овие параметри се запишуваат во калибрациската матрица K која се дефинира како:

$$\begin{bmatrix} f_x & 0 & 0 \\ s & f_y & 0 \\ c_x & c_y & 1 \end{bmatrix} \quad (170)$$



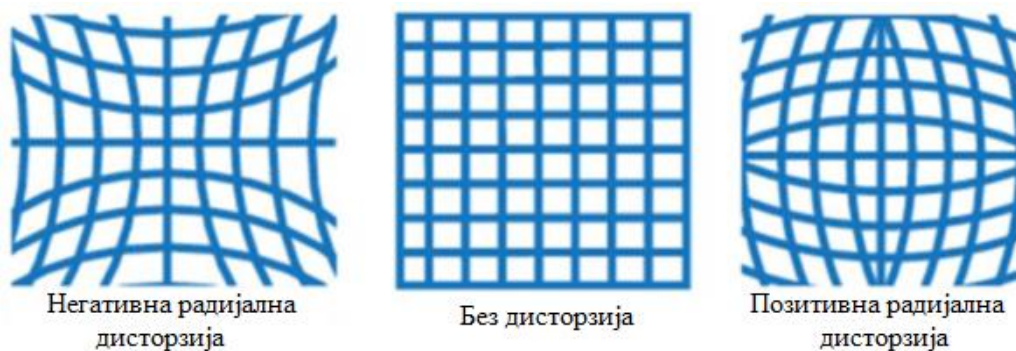
Слика 58. Закосеност на пиксел.

Зависностите на параметрите од калибрациската матрица се дадени во следната табела:

$[c_x, c_y]$	Оптички центар, во пиксели
(f_x, f_y)	Фокална должина во пиксели
$f_x = F/p_x$	
$f_y = F/p_y$	
F	Фокална должина во единица мерка, обично милиметри
(p_x, p_y)	Големина на пиксел во единица мерка, обично милиметри
$s = f_y \tan \alpha$	Коефициент на закосеност, кој не е нула ако оските на сликата не се нормални

Табела 2. Зависност на параметри на калибрациската матрица

Радијалната дисторзија се случува кога светлосните зраци повеќе се прекршуваат блиску до краевите на сликата отколку на центарот. Колку е помала леќата, толку дисторзијата е поголема.



Слика 59. Типови на радијална дисторзија.

Дисторзираните точки се означени како $(x_{distorted}, y_{distorted})$ во формулите:

$$x_{distorted} = x(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6) \quad (171)$$

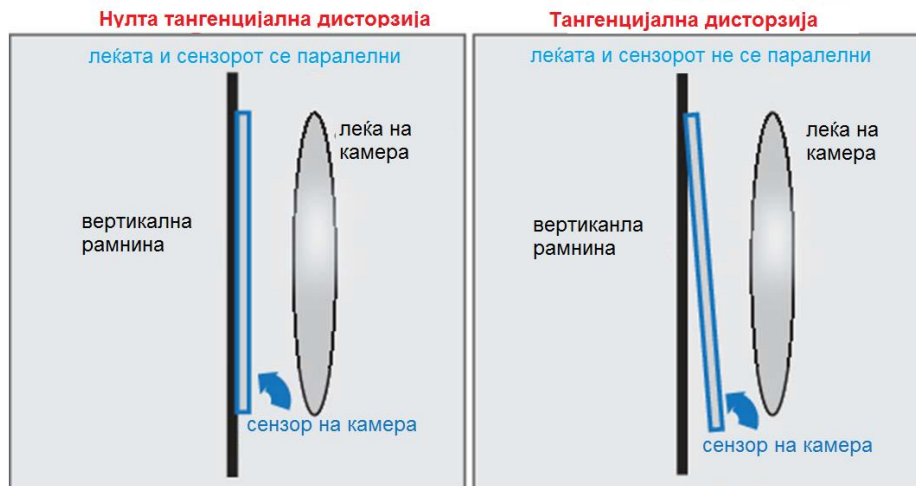
$$y_{distorted} = y(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6) \quad (172)$$

Каде што x и y се локациите на пикселите кои се недисторзирани, а k_1, k_2 и k_3 се коефициентите на радијална дисторзија на леќата.

$$r^2: x^2 + y^2 \quad (173)$$

Доволни се два коефициенти за калибрација. За голема дисторзија како кај леќите со позитивна радијална дисторзија, потребно е да се одберат 3 коефициенти што ќе ги опфатат k . Локациите на недисторзираните пиксели се нормализирани како координати на слика со центар во оптичкиот центар.

Тангенцијална дисторзија се јавува кога леќата и рамнината на сликата не се паралелни. Коефициентите на тангенцијална дисторзија ги оформуваат следниве типови на дисторзија.



Слика 60. Типови на тангенцијална дисторзија.

Точките на дисторзија се означени како $(x_{distorted}, y_{distorted})$ во формулите,

$$x_{distorted} = x + [2 * p_1 * y + p_2 * (r^2 + 2 * x^2)] \quad (174)$$

$$y_{distorted} = y + [p_1 * (r^2 + 2 * y^2) + 2 * p_2 * x] \quad (175)$$

Каде што x и y се локациите на пикселите кои се недисторзирани, а p_1 и p_2 се коефициентите на тангенцијалната дисторзија на леќата.

$$r^2 = x^2 + y^2 \quad (176)$$

Локациите на недисторзираните пиксели се нормализирани како координати на слика со центар во оптичкиот центар.

5. Методи на оптимизација базирани на градиент

Оптимизацијата се однесува на проблемот на избирање на множество на параметри со кои се врши максимизирање или минимизирање на дадена функција $f(x)$. Параметарот x припаѓа на конкретен домен D , кој често е отворено множество во \mathbb{R}^n . Во пракса, функцијата $f(\cdot)$ се нарекува објективна функција и често е претставена како грешка, добивка или услов кој треба да биде оптимизиран. Изборот на објективна функција често е одреден со статистички или геометриски модел, кој што се користи за пресметување на грешката при мерења [1]. На пример, добивањето на оптималното решение x^* преку минимизирање на објективната функција $f(\cdot)$, се нарекува *неограничена оптимизација*

$$x^* = \underset{x}{\operatorname{argmin}} f(x), \quad x \in \mathbb{R}^n, \quad (177)$$

каде $f(\cdot): \mathbb{R}^n \rightarrow \mathbb{R}$ е најмалку двојно диференцијабилна во \mathbb{R}^n или формално $f(\cdot) \in C^2(\mathbb{R}^n)$.

Бидејќи $f(\cdot)$ е двојно диференцијабилна, често го дефинираме нејзиниот градиент како $x = [x_1, x_2, \dots, x_n]^T \in \mathbb{R}^n$, означен со $\nabla f(x)$, за да биде вектор

$$\nabla f(x) \doteq \left[\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \dots, \frac{\partial f(x)}{\partial x_n} \right]^T \in \mathbb{R}^n, \quad (178)$$

а воедно да се дефинира и неговата Хесиан матрица во x , означена со $\nabla^2 f(x)$, да биде

$$\nabla^2 f(x) \doteq \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2 \partial x_2} & \dots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \dots & \frac{\partial^2 f(x)}{\partial x_n \partial x_n} \end{bmatrix} \in \mathbb{R}^{n \times n}. \quad (179)$$

Забележуваме дека векторот на градиентот секогаш е насочен во спротивна насока од најголемото намалување, а Хесиан матрицата е секогаш симетрична.

5.1 Услови за оптималност

Претпоставка 1.(Задолжителни услови за оптималност). Ако вектор x^* е неограничен локален минимум на $f(\cdot)$ тогаш

$$\nabla f(x^*) = 0, \nabla^2 f(x^*) \geq 0. \quad (180)$$

Претпоставка 2.(доволни услови од втор ред за оптималност). Ако вектор $x^* \in \mathbb{R}^n$ ги задоволува условите

$$\nabla f(x^*) = 0, \nabla^2 f(x^*) > 0 \quad (181)$$

тогаш x^* претставува строго неограничен минимум на $f(\cdot)$.

Па оттука може да се каже дека во локален минимум x^* , градиентниот вектор $\nabla f(x^*)$ мора да се отстрани, а Хесиан матрицата $\nabla^2 f(x^*)$ е типично позитивно дефинирана [1].

Локалниот минимум не претставува задолжително глобален минимум што го баравме. Сепак условите кои се дадени погоре се само локален критериум и поради тоа може да се разликува локалниот од глобалниот минимум. Во специјален случај кога функцијата $f(\cdot)$ е конвексна, тогаш има само еден минимум и поради тоа локалниот минимум мора да биде и глобален.

5.2 Алгоритми

Основната идеја за минимизирање на нелинеарна функција $f(\cdot)$ е доста едноставна. Почнуваме со почетна претпоставка $x = x^0$, а потоа постепено се надградува x во x^1, x^2, \dots , на таков начин да величината $f(x)$ се намалува за секоја итерација, т.е. $f(x^{i+1}) \leq f(x^i)$. Се разбира, најбезбедниот начин за да се обезбеди намалување на вредноста е да се следи насоката на намалување, која што во нашиот случај би била спротивната насока на градиент векторот. Оваа идеја додава пораст на методот на најголемо намалување за барање на минимумот. На секоја итерација,

$$x^{i+1} = x^i - \alpha^i \nabla f(x^i), \quad (182)$$

за одреден скалар α^i , наречен големина на чекор. Постојат многу различни избори за големината на чекорот α^i , а наједноставниот избор е секако да се постави како мала константа.

Иако векторот $-\nabla f(x^i)$ е насочен кон насоката на најголемото намалување, локално околу x^i , не мора да значи дека претставува најдобриот избор за пребарување на минимумот при поголем размер. Модификација на горе наведениот метод на градиент е во форма

$$x^{i+1} = x^i - \alpha^i D^i \nabla f(x^i), \quad (183)$$

каде што $D^i \in \mathbb{R}^{n \times n}$ е позитивно дефинирана симетрична матрица за да биде пресметана во секој алгоритам. Методот на најголемото намалување прераснува во специјален случај каде $D^i \equiv I$. Во главно, D^i може да се види како тежинска матрица, која што ја подесува насоката на намалување согласно на подобрената локална информација за $f(\cdot)$, отколку само градиентот. Едноставен избор за D^i би била дијагонална матрица што ја скалира брзината на намалувањето различно во секоја насока. Њутновиот метод подолу е класичен пример за подобрен избор за D^i .

Њутнов метод

Задолжителните и доволните услови за оптималност сугерираат дека околу локален минимум, функцијата $f(\cdot)$ приближно разложува квадратна функција

$$f(x) \approx f(x^i) + \nabla f(x^i)^T (x - x^i) + \frac{1}{2} (x - x^i)^T \nabla^2 f(x^i) (x - x^i). \quad (184)$$

Ова покажува дека кога x^i е блиску до минимумот x^* или функцијата $f(\cdot)$ потребна е квадратна функција, минимумот x^* е најдобро пресметан со x што прави отстранување на равенката од десната страна,

$$\nabla f(x^i) + \nabla^2 f(x^i) (x - x^i) = 0. \quad (185)$$

со ова се добива Њутновиот метод

$$x^{i+1} = x^i - \left(\nabla^2 f(x^i) \right)^{-1} \nabla f(x^i). \quad (186)$$

Поопшта итерација користи

$$x^{i+1} = x^i - \alpha^i \left(\nabla^2 f(x^i) \right)^{-1} \nabla f(x^i), \quad (187)$$

Каде големината на чекорот α^i се користи за контрола на брзината на напредувањето. Забележуваме дека Њутновиот метод е само специјален случај од општиот метод на итерација равенка (183) со $D^i = (\nabla^2 f(x^i))^{-1}$.

Гаус-њутнов метод и левенберг-маркارتов метод

Пресметката на Хесиан матрицата $\nabla^2 f(x)$ е честопати скапа и понекогаш невозможна, т.е. функцијата $f(\cdot)$ не е двојно диференцијабилна. Во таков случај алтернативи на D^i се прилагодени така да при одредени проширувања ја пресметаат матрицата $(\nabla^2 f(x^i))^{-1}$ и користат само информацијата добиена во првиот диференцијал на $f(x)$. Гаус-њутновиот метод е токму таков метод, што се применува само кај проблемот при минимизирање на сумата од квадрати на функции со реални вредности, $u(x) = [u_1(x), u_2(x), \dots, u_m(x)]^T$. За векторска функција $u(\cdot)$ ја дефинираме нејзината јакобиева матрица како

$$\nabla u(x) \doteq \begin{bmatrix} \frac{\partial u_1(x)}{\partial x_1} & \frac{\partial u_1(x)}{\partial x_2} & \dots & \frac{\partial u_1(x)}{\partial x_n} \\ \frac{\partial u_2(x)}{\partial x_1} & \frac{\partial u_2(x)}{\partial x_2} & \dots & \frac{\partial u_2(x)}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial u_m(x)}{\partial x_1} & \frac{\partial u_m(x)}{\partial x_2} & \dots & \frac{\partial u_m(x)}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{m \times n}. \quad (188)$$

Типично, $m \gg n$, а Јакобиевата матрица е секогаш од цел ранг n . Па за објективната функција $f(\cdot)$ сега во формата

$$f(x) = \frac{1}{2} \|u(x)\|^2 = \frac{1}{2} \sum_{i=1}^m u_i(x)^2, \quad (189)$$

наместо да се избере $D^i = \nabla^2 f(x^i)$, Гаус-њутновиот метод го избира

$$D^i = (\nabla u(x^i) \nabla u(x^i)^T)^{-1}. \quad (190)$$

Бидејќи $\nabla f(x) = \nabla u(x) u(x)$, Гаус-њутновата итерација има форма

$$x^{i+1} = x^i - \alpha^i (\nabla u(x^i) \nabla u(x^i)^T)^{-1} \nabla u(x^i) u(x^i). \quad (191)$$

Гаус-њутновиот метод е приближување кон Њутновиот метод, посебно кога вредноста $\|u(x)\|^2$ е мала. Секако овде користиме стандардна двојна нормализација за да се измери $u(x)$. Потребно е методот минимално да се

промени кога различна квадратна нормализација се користи, да кажеме $\|u(x)\|_Q = u(x)^T Q u(x)$ за позитивно дефинирана симетрична матрица $Q \in \mathbb{R}^{m \times m}$.

Левенберг-Маркартовиот метод всушност претставува мала модификација на Гаус-Њутновиот метод, кој што многу се користи во литературата за компјутерска визија [1]. Единствената разлика од Гаус-Њутновиот метод е да се постави $\alpha^i = 1$ и да се употреби наместо

$$D^i = (\lambda^i I + \nabla u(x^i) \nabla u(x^i)^T)^{-1} \in \mathbb{R}^{n \times n}, \quad (192)$$

Каде што $\lambda^i > 0$ е скаларно детерминиран преку следниве правила: иницијално се дадени мали вредности, а потоа:

- 1) Ако моменталната вредност на λ резултира со намалување на грешката, тогаш итерацијата е прифатена, а λ е поделена со 10 и е како иницијална вредност за наредната итерација.
- 2) Ако λ резултира со зголемување на грешката, тогаш таа е помножена со 10, па итерацијата се повторува се додека не се добие λ таква што ќе резултира со намалување на грешката.

Поради изборот на D^i во равенката (5.15), се гледа дека левенберг-маркартовиот метод сеуште работи иако понекогаш јакобиевата матрица $\nabla u(x)$ не е од целосен ранг, што често се случува во пракса. Алгоритмот тежнее да ја прилагоди големината на чекорот (преку контрола на вредноста на λ) базирана на историјата на вредностите на објективната функција $f(x)$.

Избор на големината на чекорот

Во чистиот Њутнов метод и Левенберг-Маркартовиот метод метод, не мора да се специфира големината на чекорот α^i . Во други методи пак мора да се знае како да се избере правилно големината на чекорот α^i . Наједноставниот начин е да се избере големината на чекорот да е константа, но така да не секогаш резултира со намалување на вредноста на $f(x)$ при секоја итерација. Па отука α^i често се избира да биде вредност α^* , што ќе биде добиена со решавање на линија на минимизирачки проблем:

$$\alpha^* = \arg \min_{\alpha \geq 0} f(x^i - \alpha D^i \nabla f(x^i)).$$

Ова се нарекува правило на минимизирање. Но сепак пронаоѓањето на оптимален α^* при секоја итерација претставува пресметувачки трошок. Популарен избор на α^i , без решавање на минимизирачки проблем, но сепак обезбедувајќи конвергенција е Армијо правилото [1]: за префиксни скалари s, β и σ , со $0 < \beta, \sigma < 1$, подесуваме $\alpha^i = \beta^{k_i} s$, каде k_i е првата ненегативна вредност k за која

$$f(x^i) - f(x^i - \beta^k s D^i \nabla f(x^i)) \geq -\sigma \beta^k s \nabla f(x^i)^T D^i \nabla f(x^i). \quad (193)$$

Типичен избор за скаларите е $\sigma \in [10^{-5}, 10^{-1}]$, $\beta \in [0.1, 0.5]$, $s = 1$.

6. Одредување на ориентација со помош на векторски мерења од единечна видео камера

6.1 Геометриски модел на единечна камера

Ориентација на круто тело во просторот е претставена како тековна ориентација на еден координатен систем (координатен систем врзан за телото) во однос кон друг координатен систем (фиксен координатен систем), на пример Земјен координатен систем или координатен систем врзан за точка на Земјата. Оските на двата координатни системи се поврзани преку линеарна трансформација, најчесто тоа е матрица на трансформација. Оваа матрица може да биде претставена од повеќе групи на координати, како што се: Ојлерови агли, кватерниони итн, за ова повеќе во Додаток 1.

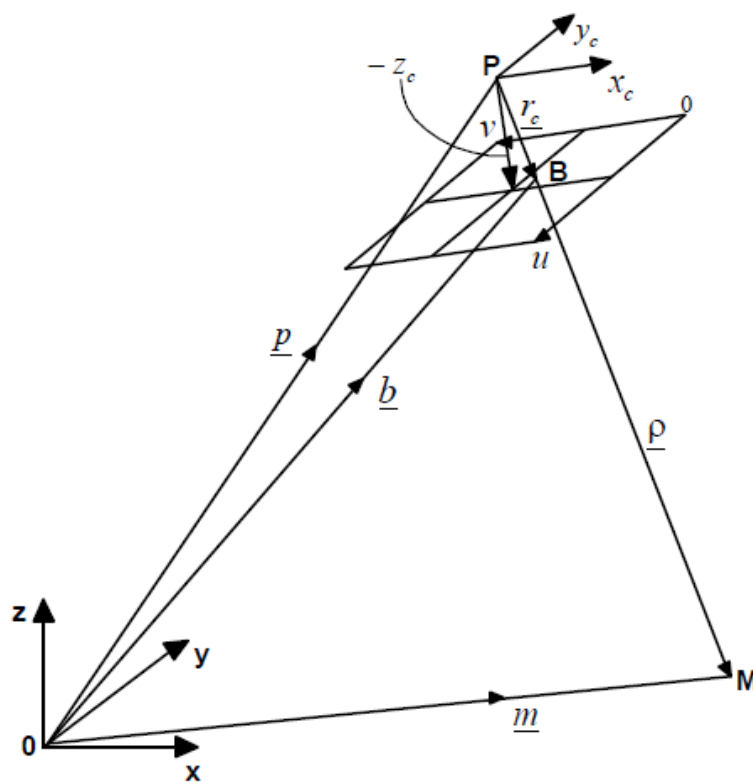
Употребата на кватерниони е најчесто применуван, поради линеарноста на кватернионите, избегнувањето на тригонометриските функции и малиот број на параметри за успешна имплементација. Единствената негативна страна на кватернионите е тоа што немаат едноставна геометриска репрезентација, па заради тоа не можат да бидат измерени директно [4].

Додека пак проблемот со мерење добиен од единечна камера е во тоа што немаме точен податок за длабочината. Па поради тоа во нашиов случај тоа растојание ќе го земеме како познато и константно.

6.2 Равенки за одредување на ориентација

Имаме камера монтирана на тело на таков начин да координатниот систем на камерата ($P; x_c, y_c, z_c$) се наоѓа во центарот на гравитација на телото, со што воедно претставува истиот координатен систем на телото. Центарот на координатниот систем на камерата е означен со P . Оската x_c е дефинирана во насока што покажува напред и е нормална на хоризонталната компонента u од сликата на камерата. Оската z_c е нормална на оската x_c и е еднаква на фокалното растојание на камерата.

Негативниот знак се појавува пред z_c , кога z_c е насочен кон центарот на сликата. Оската y_c го комплетира координатниот систем за кој важи правилото на десна рака, а воедно е и нормална на вертикалната компонента v од сликата 61.



Слика 61. Модел на единечна камера (векторска апликација)

Во координатниот систем на камерата го пресметуваме векторот на камерата \underline{r}_c . Единечниот вектор со насока на \underline{r}_c е означен како $\hat{\underline{r}}_c$ и може да се пресмета како

$$\hat{\underline{r}}_c = \frac{\underline{r}_c}{|\underline{r}_c|}. \quad (194)$$

Векторот \underline{r}_c може да се трансформира во координатен систем врзан за точка на Земјата т.н. навигациски координатен систем преку матрица на трансформирање C_{b2n} , односно тоа е матрица со која што координатите од координатниот систем на камерата ги трансформира во координати во навигацискиот координатен систем:

$$\underline{r} = C_{b2n} \underline{r}_c \quad (195)$$

Каде што C_{b2n} претставен во форма на кватерниони изгледа вака:

$$C_{b2n} = \begin{bmatrix} (q_0^2 + q_1^2 - q_2^2 - q_3^2) & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & (q_0^2 - q_1^2 + q_2^2 - q_3^2) & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & (q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{bmatrix}. \quad (196)$$

После трансформацијата за векторот \underline{r} важи следново [5]:

- а) $\underline{r} = \underline{b} - \underline{p}$
- б) $\hat{\underline{r}} = \frac{\underline{b} - \underline{p}}{|\underline{b} - \underline{p}|}$, единечниот вектор ја прикажува насоката.
- в) $\underline{\rho} = \mu \hat{\underline{r}}$, каде што опсегот μ е непознат параметар.

Длабочината $\underline{\rho}$ претставува разликата помеѓу векторот \underline{m} и векторот \underline{p} и може да се запише како:

$$\underline{\rho} = \underline{m} - \underline{p} \quad (197)$$

Единечниот вектор во насоката $\underline{\rho}$ претставен како $\hat{\underline{r}}$ во навигацискиот координатен систем исто така може да се пресмета како:

$$\hat{\underline{r}} = \frac{\underline{\rho}}{|\underline{\rho}|}. \quad (198)$$

Векторот кватернион $\underline{q} = [q_0 \ q_1 \ q_2 \ q_3]^T$ користен во матрицата на трансформирање (196) е со нормализација, т.е.

$$|\underline{q}| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1. \quad (199)$$

Добрата особина на кватернионите е што можат да се искористат за трансформација на единечниот вектор $\hat{\underline{r}}_c$ во координатниот систем на камерата, во единечен вектор $\hat{\underline{r}}$ во навигациониот координатен систем:

$$\hat{\underline{r}} = C_{b2n}\hat{\underline{r}}_c. \quad (200)$$

Претпоставуваме дека ги имаме податоците за векторот \underline{m} , векторот $\underline{\rho}$ од каде се врши мерењето и монтираната единечна камера [5]. Вршиме мерења и снимања со камерата од точката и сакаме да ја одредиме ориентацијата на камерата во зависност од околината, како на слика 61.

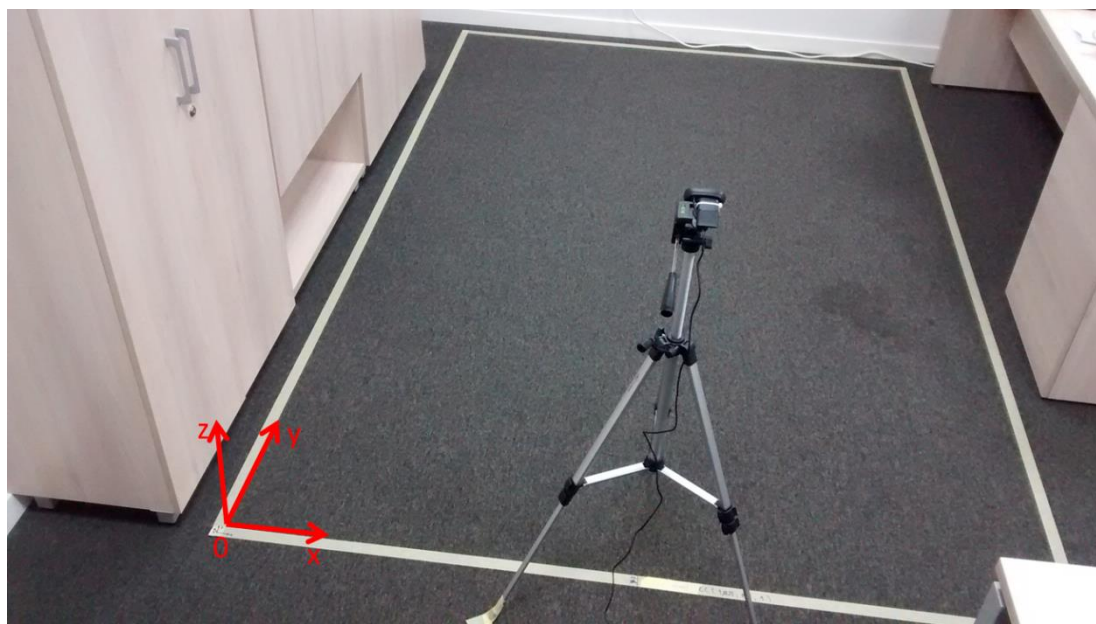
Всушност сакаме да го пресметаме векторот од кватерниони $\underline{q} = [q_0 \ q_1 \ q_2 \ q_3]^T$. Со мерењата и пресметките од камерата се добива единечниот вектор $\hat{\underline{r}}_c$ во координатниот систем на камерата, равекната (194). Користејќи ги векторите \underline{m} и $\underline{\rho}$ ја пресметуваме длабочината $\underline{\rho}$ преку равенката (197). Следно ја користиме длабочината $\underline{\rho}$ во равенката (198) за да го пресметаме единечниот вектор $\hat{\underline{r}}$ во навигацискиот координатен систем. Сега откако ги добивме и двата единечни вектора, едниот во координатниот систем на камерата, а другиот во навигацискиот координатен систем, може да го одредиме векторот на кватерниони \underline{q} . Равенката (200) во матрична форма е:

$$\begin{bmatrix} \hat{\underline{r}}_x \\ \hat{\underline{r}}_y \\ \hat{\underline{r}}_z \end{bmatrix} = \begin{bmatrix} (q_0^2 + q_1^2 - q_2^2 - q_3^2) & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & (q_0^2 - q_1^2 + q_2^2 - q_3^2) & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & (q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{bmatrix} \begin{bmatrix} \hat{\underline{r}}_{cx} \\ \hat{\underline{r}}_{cy} \\ \hat{\underline{r}}_{cz} \end{bmatrix} \quad (201)$$

Бидејќи векторот на кватерниони \underline{q} претставува вектор со четири елементи, дополнителна равенка може да се додаде на равенката (201). Особината на векторот на кватерниони кога е нормализиран, т.е. равенката (199) може да се додаде на (201).

7. Практични експерименти и резултати

Оформена е работна околина за практичните експерименти со следниве димензии: 2 метри по x-оска, 3 метри по y-оска и 0 метри по z-оска. Работната околина и координатниот почеток на навигацискиот координатен систем се прикажани на слика 62.



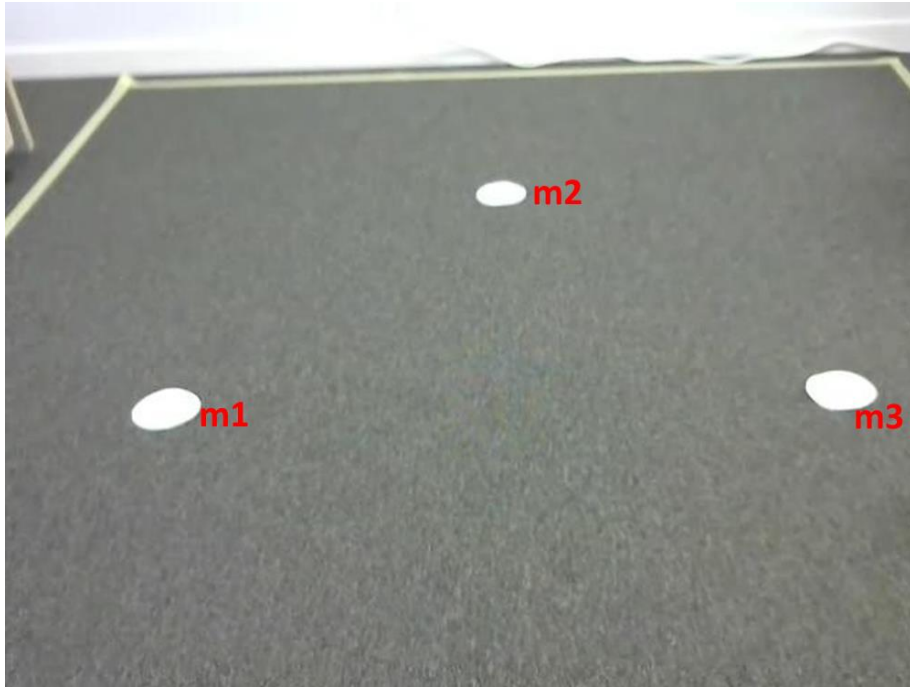
Слика 62. Работната околина и навигациски координатен систем

Камера Logitech C270 е поставена на стенд за камера. Камерата има фокална должина од 4mm. Со калибрација на камерата добиена е големина од 0.0070622mm за секој единечен пиксел. Со камера Logitech C270 снимани се три кратки видеа на фреквенција од 15Hz со должина од 20 секунди со резолуција од 640×480 пиксели. Во првото видео се менува аголот на скршнување на камерата, во второто видео се менува аголот на извишување на камерата, додека во третото видео се менува аголот на валање на камерата.

Со прецизни мерења добиен е почетниот вектор на позиција на камерата $\underline{p} = [1.0285m \ 0.05m \ 1m]^T$. Почетните Ојлерови агли дадени во степени прецизно се одредени: агол на валање $\phi = 0$, агол на извишување $\theta = -3.5$ и агол на скршнување $\psi = 0$.

Три мали кругови со радиус од 50mm се поставени на работната околина. Тие ќе послужат како точки за одредување на ориентацијата на камерата. Позицијата

на секоја од точките е позната и следна: $\underline{m}_1 = [0.5m \ 1.5m \ 0m]^T$, $\underline{m}_2 = [1m \ 2.25m \ 0m]^T$ и $\underline{m}_3 = [1.5m \ 1.5m \ 0m]^T$. Круговите се прикажани на слика 63.



Слика 63. Три кругови поставени на работната околина

7.1 Примена на Гаус-Њутновиот метод и Левенберг-Маркартовиот метод за одредување на ориентација

Пакетот во Матлаб наречен Optimization toolbox содржи готови функции за оптимизација. Во нашиот магистерски труд ние ја користиме функцијата lsqnonlin. Оваа функција овозможува решавање на проблеми како нелинеарни најмали квадрати (анг. nonlinear least-squares problems) и проблеми за нелинеарно совпаѓање на податоци (анг. nonlinear data-fitting problems).

Функцијата lsqnonlin бара дефинирање на кориснички функции за оредување на параметри во векторска форма. Ова е генералната форма за оптимизациски проблем кој може да се реши со lsqnonlin:

$$F(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ f_3(x) \end{bmatrix} \quad (202)$$

каде x е вектор или матрица и $f(x)$ е функција/функции кои враќаат назад векторски или матрични вредности. На пример ако напишеме во Матлаб

```
x = lsqnonlin(fun,x0)
```

lsqnonlin започнува од точка x_0 и го наоѓа решението опишано со функцијата `fun`. Со дефинирање на опцијата `options = optimset('LevenbergMarquardt', 'on');` се избира Левенберг-Маркартовиот метод вредност со $\lambda = 0.01$. Со дополнително подесување на `options.LevenbergMarquardt` во 'off' и `options.LargeScale` во 'off' се избира Гаус-њутновиот метод.

За нашиот магистерски труд, за решавање на равенката (201) со трите познати точки во работната околина ја дефиниравме следнава Матлаб функција:

```
function y =master_attitide_gnewton(ro1x, ro1y, ro1z, ro2x, ro2y, ro2z,
ro3x, ro3y, ro3z, rx1, ry1, rz1, rx2, ry2, rz2, rx3, ry3, rz3, a, b, c, d);

x0=[a; b; c; d];
% options = optimset('LevenbergMarquardt', 'on');
% options = optimset('LargeScale', 'off', 'LevenbergMarquardt', 'off');
% [y] = fsolve(@mojafun, x0, options);
% [y] = fsolve(@mojafun, x0);

[y] = lsqnonlin(@mojafun,x0);

function F = mojafun(x)

F = [rx1-((x(1)^2+x(2)^2-x(3)^2-x(4)^2)*ro1x)-(2*(x(2)*x(3)-
x(1)*x(4))*ro1y)-(2*(x(2)*x(4)+x(1)*x(3))*ro1z)

      ry1-(2*(x(2)*x(3)+x(1)*x(4))*ro1x)-((x(1)^2-x(2)^2+x(3)^2-
x(4)^2)*ro1y)-(2*(x(3)*x(4)-x(1)*x(2))*ro1z)

      rz1-(2*(x(2)*x(4)-x(1)*x(3))*ro1x)-(2*(x(3)*x(4)+x(1)*x(2))*ro1y)-
((x(1)^2-x(2)^2-x(3)^2+x(4)^2)*ro1z)

      rx2-((x(1)^2+x(2)^2-x(3)^2-x(4)^2)*ro2x)-(2*(x(2)*x(3)-
x(1)*x(4))*ro2y)-(2*(x(2)*x(4)+x(1)*x(3))*ro2z)

      ry2-(2*(x(2)*x(3)+x(1)*x(4))*ro2x)-((x(1)^2-x(2)^2+x(3)^2-
x(4)^2)*ro2y)-(2*(x(3)*x(4)-x(1)*x(2))*ro2z)

      rz2-(2*(x(2)*x(4)-x(1)*x(3))*ro2x)-(2*(x(3)*x(4)+x(1)*x(2))*ro2y)-
((x(1)^2-x(2)^2-x(3)^2+x(4)^2)*ro2z)

      rx3-((x(1)^2+x(2)^2-x(3)^2-x(4)^2)*ro3x)-(2*(x(2)*x(3)-
x(1)*x(4))*ro3y)-(2*(x(2)*x(4)+x(1)*x(3))*ro3z)

      ry3-(2*(x(2)*x(3)+x(1)*x(4))*ro3x)-((x(1)^2-x(2)^2+x(3)^2-
x(4)^2)*ro3y)-(2*(x(3)*x(4)-x(1)*x(2))*ro3z)
```

$$rz3 - (2 * (x(2) * x(4) - x(1) * x(3)) * ro3x) - (2 * (x(3) * x(4) + x(1) * x(2)) * ro3y) -$$

$$((x(1)^2 - x(2)^2 - x(3)^2 + x(4)^2) * ro3z)];$$

Забележуваме во кодот погоре дека имаме дефинирано точно 9 равенки, за секоја точка во работната околина по три равенки, согласно равенката (201).

Комплетниот Матлаб код за одредување на ориентацијата на камерата во форма на кватерниони е следниот:

```

RGB = imread('image2.avi_0060.jpg'); %read image
I = rgb2gray(RGB); %Intensity
bw =im2bw(I, 0.81); %Black and White

% remove all objects containing fewer than 30 pixels
bw = bwareaopen(bw,30);

% fill a gap in the pen's cap
se = strel('disk',2);
bw = imclose(bw,se);

% fill any holes, so that regionprops can be used to estimate
% the area enclosed by each of the boundaries
bw = imfill(bw,'holes');

[B,L] = bwboundaries(bw,'noholes');

no_objects_detected=length(B)

% Display the label matrix and draw each boundary
imshow(label2rgb(L, @jet, [.5 .5 .5]))
hold on
for k = 1:length(B)
    boundary = B{k};
    plot(boundary(:,2), boundary(:,1), 'w', 'LineWidth', 2)
end

% Office white wall was detected as blob. Ignore it. Take only the circular
% blobs. Estimate their center pixel coordinates pu and pv (horizontal and
% vertical respectively)

for k = 2
    boundary = B{k};
    dim = size(boundary);
    pu1=sum(boundary(:,2))/dim(1);
    pv1=sum(boundary(:,1))/dim(1);
end

for k = 3
    boundary = B{k};
    dim = size(boundary);
    pu2=sum(boundary(:,2))/dim(1);
    pv2=sum(boundary(:,1))/dim(1);
end

```

```

for k = 4
    boundary = B{k};
    dim = size(boundary);
    pu3=sum(boundary(:,2))/dim(1);
    pv3=sum(boundary(:,1))/dim(1);
end

xc1=pu1;    %pixel values blob 1
yc1=pv1;

xc2=pu2;    %pixel values blob 2
yc2=pv2;

xc3=pu3;    %pixel values blob 3
yc3=pv3;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parameters without calibration
x0=320;
y0=240;      %camera specifications x0 and y0 in pixel values
              (optical center)
f=0.004;      %focal lenght f in meters for Logitech Logitech C270

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parameters after calibration
x0=313.9181;
y0=245.5899; %camera specifications x0 and y0 in pixel values
              (optical center)
f=0.004;      %focal lenght f in meters for Logitech C270

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
r1x=xc1-x0;
r1y=yc1-y0;   %relative camera coordinates from the optical center
r1z=-f;

r2x=xc2-x0;
r2y=yc2-y0;   %relative camera coordinates from the optical center
r2z=-f;

r3x=xc3-x0;
r3y=yc3-y0;   %relative camera coordinates from the optical center
r3z=-f;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

bclx=r1x*0.0000070622; %the relative camera coordinates transformed from
pixels to meters
bcly=r1y*0.0000070622;
bclz=r1z;

rolx=bclx/sqrt(bclx^2+bcly^2+bclz^2); % (normalization) the unit direction
vector components
roly=bcly/sqrt(bclx^2+bcly^2+bclz^2);
rolz=bclz/sqrt(bclx^2+bcly^2+bclz^2);

bc2x=r2x*0.0000070622; %the relative camera coordinates transformed from
pixels to meters
bc2y=r2y*0.0000070622;
bc2z=r2z;

```



```

ro2x=bc2x/sqrt(bc2x^2+bc2y^2+bc2z^2); % (normalization) the unit direction
vector components
ro2y=bc2y/sqrt(bc2x^2+bc2y^2+bc2z^2);
ro2z=bc2z/sqrt(bc2x^2+bc2y^2+bc2z^2);

bc3x=r3x*0.0000070622; %the relative camera coordinates transformed from
pixels to meters
bc3y=r3y*0.0000070622;
bc3z=r3z;

ro3x=bc3x/sqrt(bc3x^2+bc3y^2+bc3z^2); % (normalization) the unit direction
vector components
ro3y=bc3y/sqrt(bc3x^2+bc3y^2+bc3z^2);
ro3z=bc3z/sqrt(bc3x^2+bc3y^2+bc3z^2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

mx1=0.5;
my1=1.5; %true(known) map point 1 position components(meters)
mz1=0;

M1=[mx1 my1 mz1]';

mx2=1;
my2=2.25; %true(known) map point 2 position components(meters)
mz2=0;

M2=[mx2 my2 mz2]';

mx3=1.5;
my3=1.5; %true(known) map point 3 position components(meters)
mz3=0;

M3=[mx3 my3 mz3]';

xc=1.0285;
yc=0.05; %true(known) camera position components(meters)
zc=1;

P=[xc yc zc]';

ro1=M1-P; %M1 range components

rx1=ro1(1)/sqrt(ro1(1)^2+ro1(2)^2+ro1(3)^2);
ry1=ro1(2)/sqrt(ro1(1)^2+ro1(2)^2+ro1(3)^2); % M1 unit direction
components
rz1=ro1(3)/sqrt(ro1(1)^2+ro1(2)^2+ro1(3)^2);

ro2=M2-P; %M2 range components

rx2=ro2(1)/sqrt(ro2(1)^2+ro2(2)^2+ro2(3)^2);
ry2=ro2(2)/sqrt(ro2(1)^2+ro2(2)^2+ro2(3)^2); % M2 unit direction
components
rz2=ro2(3)/sqrt(ro2(1)^2+ro2(2)^2+ro2(3)^2);

ro3=M3-P; %M3 range components

```

```

rx3=ro3(1)/sqrt(ro3(1)^2+ro3(2)^2+ro3(3)^2);
ry3=ro3(2)/sqrt(ro3(1)^2+ro3(2)^2+ro3(3)^2); % M2 unit direction
components
rz3=ro3(3)/sqrt(ro3(1)^2+ro3(2)^2+ro3(3)^2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
q0 = 0.999533591200424;
q1 = 0; % initial quaternion roll=0, pitch=-3.5, yaw=0
q2 = -0.030538501305477;
q3 = 0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
y=master_attitude_gnewton(rolx, roly, rolz, ro2x, ro2y, ro2z, ro3x, ro3y,
ro3z, rx1, ry1, rz1, rx2, ry2, rz2, rx3, ry3, rz3, q0, q1, q2, q3)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

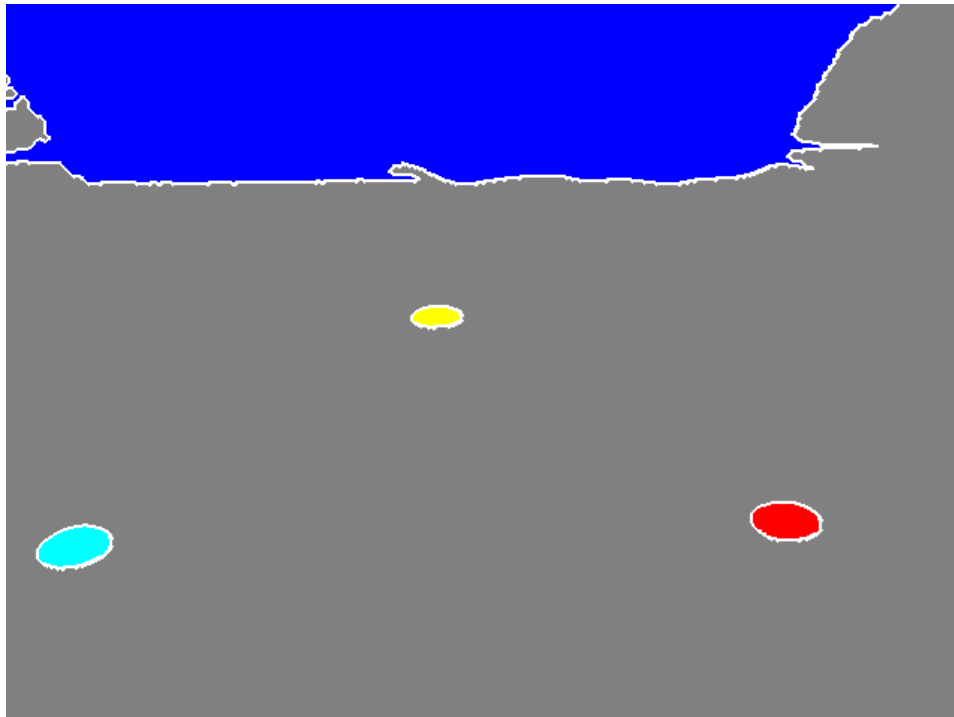
```

7.2 Резултати од практичните експерименти

Резултатите од практичните експерименти со првото видео каде се менуваше аголот на скршнување на камерата се дадени на слика 64 и 65. Земена е 60-тата слика од 4-тата секунда од видеото.



Слика 64. Промена на аголот на скршнување, пред обработка на сликата

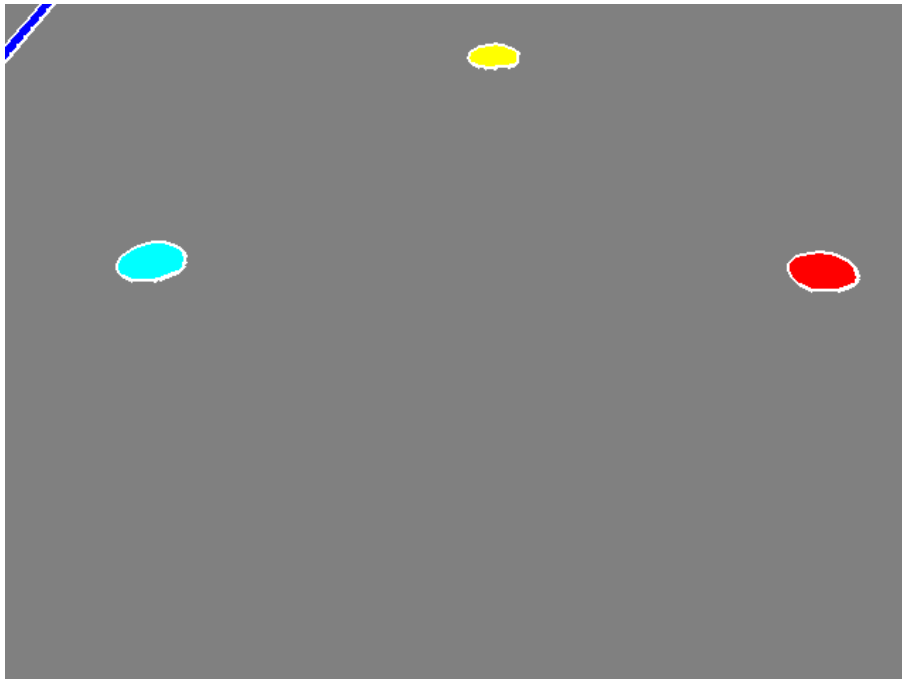


Слика 65. Промена на аголот на скршнување, со обработка на сликата

Резултатите од практичните експерименти со второто видео каде се менуваше аголот на извишување на камерата се дадени на слика 66 и 67. Земена е 180-тата слика од 12-тата секунда од видеото.



Слика 66. Промена на аголот на извишување, пред обработка на сликата

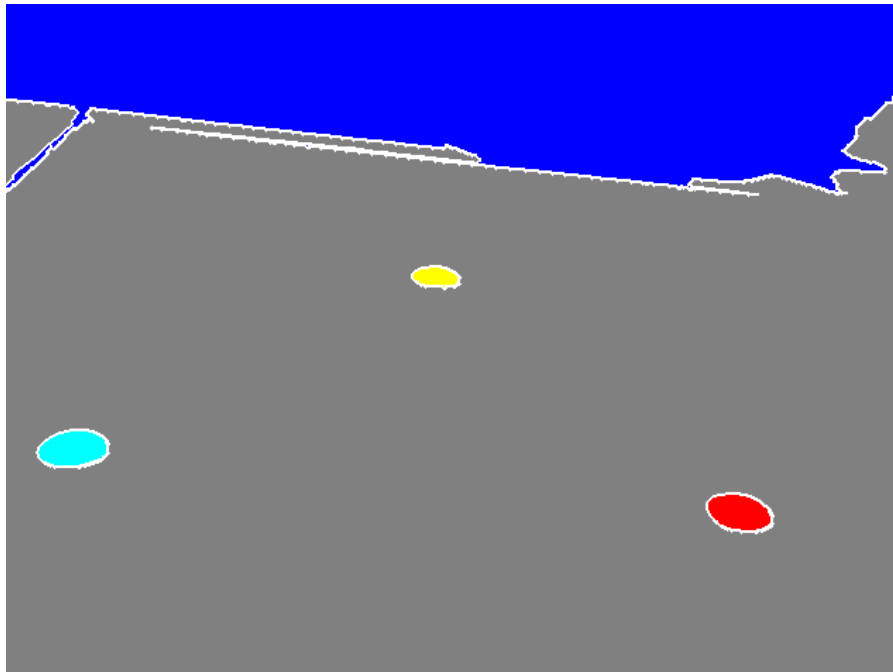


Слика 67. Промена на аголот на извишување, со обработка на сликата

Резултатите од практичните експерименти со второто видео каде се менуваше аголот на валање на камерата се дадени на слика 68 и 69. Земена е 60-тата слика од 4-тата секунда од видеото.



Слика 68. Промена на аголот на валање, пред обработка на сликата



Слика 69. Промена на аголот на валање, со обработка на сликата

Со помош на овие практични експерименти во кои што се користени двата метода, успешно се пресметани новите Ојлерови агли за позицијата на камерата, односно, со тие агли се одредува ориентацијата на камерата во однос на околината која што е поставена.

Вредности кватерниони	q0	q1	q2	q3
Слика 63 (почетна)	0.99953359 1200424	0	- 0.030538501 305477	0
Слика 65	0.9108	0.3526	-0.0063	-0.0664

Табела 3. Компарација на почетните вредности со новодобиените вредности изразени во кватерниони

Овие детални вредности, од табела 3 од експериментот, посочуваат за вредностите пред и по извршеното движење на камерата. Почетните и новите вредности изразени во кватерниони се прикажани, но сепак нивната презентација вака не претставува јасна слика за ориентацијата на камерата или пак за промените кои се имаат случено. Па затоа во табела 4 кватернионите се

претставени во форма на Ојлерови агли [Додаток 1], кои што пак се полесни за разбирање и визуелизирање.

Вредности во Ојлерови агли	Агол на скршнување	Агол на извишување	Агол на валање
Слика 63 (почетна)	0	-3.5	0
Слика 65	-0.1313	0.0370	0.7362

Табела 4. Компарација на почетните вредности со новодобиените вредности изразени во форма на Ојлерови агли

Идеата со експерименталното движење претставено на слика 64, се однесува за промени направени на ориентацијата на камерата само во однос на вертикалната оска со промена на аголот на скршнување. Резултатите добиени во табелите 3 и 4 по извршување на двата метода, Гаус-Њутновиот и Левенберг-Маркартовиот, врз слика 64 споредена со почетната слика 63, не се така едноставни. Може да се види дека со движење релативно само околу вертикалната оска со цел да се промени само аголот на скршнување, промени се направени на вредностите кај сите три Ојлерови агли. Тоа доведува до заклучок дека сите три агли мора да се земат во предвид при одредување на ориентација на тело.

8. Заклучок и работа во иднина

Одредувањето на ориентација на подвижни објекти во разни области е доста сложен проблем за решавање. Во овој магистерски труд беше презентирана рамка за одредување на ориентација на подвижен објект со помош на векторски мерења од единечна камера во веќе позната околина. Прво преку функција во Матлаб за детекција на круг, беа детектирани 3-те кругови во работната површина, чија што локација е позната. За потоа преку процес за обработка на слики, да се дојде до решенија соодветни за ориентацијата на подвижниот објект. Решенија добивме со помош на Гаус-Њутновиот метод и Левенберг-Маркартовиот метод. Употребувавме кватерниони, поради нивната линеарност

со што се избегна пресметувањето на тригонометриските функции, како и поради малиот број на параметри за успешна имплементација, со што имавме пократко време за пресметка. За појасно претставувањето на крај вредностите ги изразивме во форма на Ојлерови агли. Резултатите добиени од решенијата на двата метода беа споредени и покажуваат дека Левенберг-Маркартовиот метод преставува подобар избор за решавање на овој проблем. Резултатите кои што ги добивме преку овој метод од севкупната обработка и анализа беа подобри. Всушност Левенберг-Маркартовиот метод преставува подобрена модификација на Гаус-Њутновиот метод, кој што во денешно време наоѓа доста голема примена во литературата за компјутерска визија.

9. Користена литература

- [1] Y. Ma, S. Soatto, J. Kosecka, S. Sastry "Invitation to 3-D Vision: From images to geometric models", Springer 2010.
- [2] Weisstein, Eric W. "Euclid's Postulates." From MathWorld--A Wolfram Web Resource. <http://mathworld.wolfram.com/EuclidsPostulates.html>
- [3] R. Szeliski "Computer Vision: Algorithms and Applications" Texts in Computer Science, Springer 2011.
- [4] J.A. Farrell and M. Barth."The Global Positioning System and Inertial Navigation", The McGraw-Hill Inc., 1999.
- [5] V. Sazdovski, P. M.G. Silson, A. Tsourdos "Attitude Determination from Single Camera Vector Observations", IEEE Conference on Intelligent Systems, London, UK, 2010.
- [6] Computer Vision Toolbox in Matlab, User Manual.
- [7] Z. Mohajerani., 'Vision-based UAV pose estimation' A Thesis at Department of Electrical and Computer Engineering, Northeastern University Boston, Massachusetts 08/2008.
- [8] Loshkovska, Suzana, and Saso Koceski, eds. ICT innovations 2015: Emerging technologies for better living. Vol. 399. Springer, 2015.
- [9] Stojanov, Done, and Saso Koceski. "Topological MRI prostate segmentation method." In Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on, pp. 219-225. IEEE, 2014
- [10] Koceski, Saso, and Natasa Koceska. "Evaluation of an assistive telepresence robot for elderly healthcare." Journal of medical systems 40, no. 5 (2016): 121.
- [11] Koceski, Saso, Natasa Koceska, and Ivica Koccev. "Design and evaluation of cell phone pointing interface for robot control." International Journal of Advanced Robotic Systems 9, no. 4 (2012): 135.
- [12] Koceski, Saso, Stojanche Panov, Natasa Koceska, Pierluigi Beomonte Zobel, and Francesco Durante. "A novel quad harmony search algorithm for grid-based path finding." International Journal of Advanced Robotic Systems 11, no. 9 (2014): 144.
- [13] Koceska, Natasa, Saso Koceski, Francesco Durante, Pierluigi Beomonte Zobel, and Terenziano Raparelli. "Control architecture of a 10 DOF lower limbs exoskeleton for gait rehabilitation." International Journal of Advanced Robotic Systems 10, no. 1 (2013): 68.
- [14] Serafimov, Kire, Dimitrija Angelkov, Natasa Koceska, and Saso Koceski. "Using mobile-phone accelerometer for gestural control of soccer robots." In Embedded Computing (MECO), 2012 Mediterranean Conference on, Bar, Montenegro, pp. 140-143. 2012.

Cekova, Katerina, Natasa Koceska, and Saso Koceski. "Gesture Control of a Mobile Robot using Kinect Sensor." (2016): 251-258.

[15] Koceski, Saso, and Natasa Koceska. "Interaction between players of mobile phone game with augmented reality (AR) interface." In 2011 International Conference on User Science and Engineering (i-USEr), pp. 245-250. IEEE, 2011.

[16] Murphy-Chutorian, Erik, and Mohan Manubhai Trivedi. "Head pose estimation in computer vision: A survey." *IEEE transactions on pattern analysis and machine intelligence* 31, no. 4 (2008): 607-626.

[17] Fanelli, Gabriele, Thibaut Weise, Juergen Gall, and Luc Van Gool. "Real time head pose estimation from consumer depth cameras." In *Joint Pattern Recognition Symposium*, pp. 101-110. Springer, Berlin, Heidelberg, 2011.

[18] Ansar, Adnan, and Konstantinos Daniilidis. "Linear pose estimation from points or lines." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25, no. 5 (2003): 578-589.

[19] Quan, Long, and Zhongdan Lan. "Linear n-point camera pose determination." *IEEE Transactions on pattern analysis and machine intelligence* 21, no. 8 (1999): 774-780.

[20] Schweighofer, Gerald, and Axel Pinz. "Robust pose estimation from a planar target." *IEEE transactions on pattern analysis and machine intelligence* 28, no. 12 (2006): 2024-2030.

[21] Bleser, Gabriele, Harald Wuest, and Didier Stricker. "Online camera pose estimation in partially known and dynamic scenes." In *2006 IEEE/ACM International Symposium on Mixed and Augmented Reality*, pp. 56-65. IEEE, 2006.

[22] Shotton, Jamie, Ross Girshick, Andrew Fitzgibbon, Toby Sharp, Mat Cook, Mark Finocchio, Richard Moore et al. "Efficient human pose estimation from single depth images." *IEEE transactions on pattern analysis and machine intelligence* 35, no. 12 (2012): 2821-2840.

[23] Kerl, Christian, Jürgen Sturm, and Daniel Cremers. "Robust odometry estimation for RGB-D cameras." In *2013 IEEE International Conference on Robotics and Automation*, pp. 3748-3754. IEEE, 2013.

[24] Frahm, Jan-Michael, Kevin Köser, and Reinhard Koch. "Pose estimation for multi-camera systems." In *Joint Pattern Recognition Symposium*, pp. 286-293. Springer, Berlin, Heidelberg, 2004.

[25] Frahm, Jan-Michael, Kevin Köser, and Reinhard Koch. "Pose estimation for multi-camera systems." In *Joint Pattern Recognition Symposium*, pp. 286-293. Springer, Berlin, Heidelberg, 2004.

[26] Prusak, A., O. Melnychuk, H. Roth, Ingo Schiller, and Reinhard Koch. "Pose estimation and map building with a time-of-flight-camera for robot navigation." *International Journal of Intelligent Systems Technologies and Applications* 5, no. 3/4 (2008): 355-364.

- [27] Mascaro, Ruben, Lucas Teixeira, Timo Hinzmann, Roland Siegwart, and Margarita Chli. "GOMSF: Graph-Optimization based Multi-Sensor Fusion for robust UAV pose estimation." In 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 1421-1428. IEEE, 2018.
- [28] Schneider, Johannes, Christian Eling, Lasse Klingbeil, Heiner Kuhlmann, Wolfgang Förstner, and Cyrill Stachniss. "Fast and effective online pose estimation and mapping for UAVs." In 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 4784-4791. IEEE, 2016.
- [29] Fu, Qiang, Quan Quan, and Kai-Yuan Cai. "Robust pose estimation for multirotor UAVs using off-board monocular vision." IEEE Transactions on Industrial Electronics 64, no. 10 (2017): 7942-7951.
- [30] Benini, Alessandro, Matthew J. Rutherford, and Kimon P. Valavanis. "Real-time, GPU-based pose estimation of a UAV for autonomous takeoff and landing." In 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 3463-3470. IEEE, 2016.
- [31] Patruno, Cosimo, Massimiliano Nitti, Ettore Stella, and Tiziana D'Orazio. "Helipad detection for accurate UAV pose estimation by means of a visual sensor." International Journal of Advanced Robotic Systems 14, no. 5 (2017): 1729881417731083.
- [32] Dufek, Jan, and Robin Murphy. "Visual pose estimation of USV from UAV to assist drowning victims recovery." In 2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), pp. 147-153. IEEE, 2016.
- [33] Shetty, Akshay, and Grace Xingxin Gao. "UAV Pose Estimation using Cross-view Geolocalization with Satellite Imagery." In 2019 International Conference on Robotics and Automation (ICRA), pp. 1827-1833. IEEE, 2019.
- [34] Benini, Alessandro, Matthew J. Rutherford, and Kimon P. Valavanis. "Experimental evaluation of a real-time GPU-based pose estimation system for autonomous landing of rotary wings UAVs." Control Theory and Technology 16, no. 2 (2018): 145-159.
- [35] Li, Han, and HaiBin Duan. "Verification of monocular and binocular pose estimation algorithms in vision-based UAVs autonomous aerial refueling system." Science China Technological Sciences 59, no. 11 (2016): 1730-1738.
- [36] Rozantsev, Artem, Sudipta N. Sinha, Debadeepta Dey, and Pascal Fua. "Flight dynamics-based recovery of a uav trajectory using ground cameras." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6030-6039. 2017.
- [37] Abbas, Ali, Assef Jafara, and Zouhair Dahrouja. "Real-time uav global pose estimation using 3d terrain engine." Journal of Telecommunication, Electronic and Computer Engineering (JTEC) 9, no. 1-4 (2017): 61-66.
- [38] Strohmeier, Michael, Thomas Walter, Julian Rothe, and Sergio Montenegro. "Ultra-wideband based pose estimation for small unmanned aerial vehicles." IEEE Access 6 (2018): 57526-57535.

- [39] Özaslan, Tolga, Shaojie Shen, Yash Mulgaonkar, Nathan Michael, and Vijay Kumar. "Inspection of penstocks and featureless tunnel-like environments using micro UAVs." In *Field and Service Robotics*, pp. 123-136. Springer, Cham, 2015.
- [40] Zhang, Cong, Xiaobin Xu, Yuhui Shi, Yimin Deng, Cong Li, and Haibin Duan. "Binocular Pose Estimation for UAV Autonomous Aerial Refueling via Brain Storm Optimization." In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pp. 254-261. IEEE, 2019.
- [41] Araar, Oualid, Nabil Aouf, and Ivan Vitanov. "Vision based autonomous landing of multirotor UAV on moving platform." *Journal of Intelligent & Robotic Systems* 85, no. 2 (2017): 369-384.
- [42] Izadi, Maziar, Amit K. Sanyal, Randy Beard, and He Bai. "GPS-denied relative motion estimation for fixed-wing UAV using the variational pose estimator." In *2015 54th IEEE Conference on Decision and Control (CDC)*, pp. 2152-2157. IEEE, 2015.
- [43] Shuai-Yong, Zheng, Fu Qiang, Hanna Tereshchenko, Wang Ti-Qiang, and Quan Quan. "An initial research on Ultra-wideband and Inertial Measurement Unit pose estimation for Unmanned Aerial Vehicle." In *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*, pp. 1834-1839. IEEE, 2016.
- [44] Perez-Paina, Gonzalo, Claudio Paz, Miroslav Kulich, Martin Saska, and Gastón Araguás. "Fusion of Monocular Visual-Inertial Measurements for Three Dimensional Pose Estimation." In *International Workshop on Modelling and Simulation for Autonomous Systems*, pp. 242-260. Springer, Cham, 2016.
- [45] Balamurugan, G., J. Valarmathi, and V. P. S. Naidu. "Survey on UAV navigation in GPS denied environments." In *2016 International conference on signal processing, communication, power and embedded system (SCOPEs)*, pp. 198-204. IEEE, 2016.
- [46] Cui, Jin Q., Shupeng Lai, Xiangxu Dong, and Ben M. Chen. "Autonomous navigation of UAV in foliage environment." *Journal of intelligent & robotic systems* 84, no. 1-4 (2016): 259-276.
- [47] Hood, Shannon, Kelly Benson, Patrick Hamod, Daniel Madison, Jason M. O'Kane, and Ioannis Rekleitis. "Bird's eye view: Cooperative exploration by UGV and UAV." In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 247-255. IEEE, 2017.
- [48] Oettershagen, Philipp, Thomas Stastny, Thomas Mantel, Amir Melzer, Konrad Rudin, Pascal Gohl, Gabriel Agamennoni, Kostas Alexis, and Roland Siegwart. "Long-endurance sensing and mapping using a hand-launchable solar-powered uav." In *Field and Service Robotics*, pp. 441-454. Springer, Cham, 2016.
- [49] Walter, Viktor, Martin Saska, and Antonio Franchi. "Fast mutual relative localization of uavs using ultraviolet led markers." In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1217-1226. IEEE, 2018.
- [50] Schmitz, Gabriel, Tiago Alves, Renato Henriques, Edison Freitas, and Ebrahim El'Youssef. "A simplified approach to motion estimation in a UAV using two filters." *IFAC-PapersOnLine* 49, no. 30 (2016): 325-330.

- [51] Koceski, Saso, Natasa Koceska, Pierluigi Beomonte Zobel, and Francesco Durante. "Characterization and modeling of a 3D scanner for mobile robot navigation." In 2009 17th Mediterranean Conference on Control and Automation, pp. 79-84. Ieee, 2009.
- [52] Panov, Stojanche, and Saso Koceski. "Area coverage in wireless sensor network by using harmony search algorithm." In 2014 3rd Mediterranean Conference on Embedded Computing (MECO), pp. 210-213. IEEE, 2014.
- [53] Koceski, Saso, and Natasa Koceska. "Challenges of videoconferencing distance education-a student perspective." *International Journal of Information, Business and Management* 5, no. 2 (2013): 274.
- [54] Perez-Grau, Francisco J., Ricardo Ragel, Fernando Caballero, Antidio Viguria, and Anibal Ollero. "An architecture for robust UAV navigation in GPS-denied areas." *Journal of Field Robotics* 35, no. 1 (2018): 121-145.
- [55] Amor-Martinez, Adrian, Angel Santamaria-Navarro, Fernando Herrero, Alberto Ruiz, and Alberto Sanfeliu. "Planar PØP: Feature-less pose estimation with applications in UAV localization." In 2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), pp. 15-20. IEEE, 2016.

Додаток 1. Однос помеѓу Ојлерови агли и кватерниони

Кватерниони претставени во форма на Ојлерови агли:

$$q_0 = \cos \frac{\phi}{2} \cos \frac{\theta}{2} \cos \frac{\psi}{2} + \sin \frac{\phi}{2} \sin \frac{\theta}{2} \sin \frac{\psi}{2}$$

$$q_1 = \sin \frac{\phi}{2} \cos \frac{\theta}{2} \cos \frac{\psi}{2} - \cos \frac{\phi}{2} \sin \frac{\theta}{2} \sin \frac{\psi}{2}$$

$$q_2 = \cos \frac{\phi}{2} \sin \frac{\theta}{2} \cos \frac{\psi}{2} + \sin \frac{\phi}{2} \cos \frac{\theta}{2} \sin \frac{\psi}{2}$$

$$q_3 = \cos \frac{\phi}{2} \cos \frac{\theta}{2} \sin \frac{\psi}{2} + \sin \frac{\phi}{2} \sin \frac{\theta}{2} \cos \frac{\psi}{2}$$

Ојлерови агли претставени во форма на кватерниони:

Агол на валање: $\phi = \arctan \left[\frac{2(q_2 q_3 + q_0 q_1)}{(q_0^2 - q_1^2 - q_2^2 + q_3^2)} \right]$

Агол на извишување: $\theta = \arcsin [-2(q_1 q_3 - q_0 q_2)]$

Агол на скршнување: $\psi = \arctan \left[\frac{2(q_1 q_2 + q_0 q_3)}{(q_0^2 + q_1^2 - q_2^2 - q_3^2)} \right]$

Владимиров Ангел

**Одредување на ориентација на подвижни објекти со
помош на векторски мерења од видео камера**

Универзитет „Гоце Делчев“ – Штип